Summer 2009

Geometrically Nonlinear Arc Length Sparse Finite Element Analysis and Optimal Design of Truss Structures

Glenn A. Hrinda
Old Dominion University

GEOMETRICALLY NONLINEAR ARC LENGTH SPARSE FINITE ELEMENT ANALYSIS AND OPTIMAL DESIGN OF TRUSS STRUCTURES

by

Glenn A. Hrinda, P.E.

B.S. May 1988, Old Dominion University, Norfolk, Virginia
M.S. May 1992, University of Virginia, Charlottesville, Virginia

A Dissertation Submitted to the Faculty of Old Dominion University
in Partial Fulfillment of the Requirements for the Degree
of

DOCTOR OF PHILOSOPHY

CIVIL ENGINEERING

OLD DOMINION UNIVERSITY
August 2009

Approved by:

_____
Duc T. Nguyen (Director)

_____
Gene Hou (Member)

_____
Chuh Mei (Member)

ABSTRACT

GEOMETRICALLY NONLINEAR ARC LENGTH SPARSE FINITE ELEMENT ANALYSIS AND OPTIMAL DESIGN OF TRUSS STRUCTURES

Glenn A. Hrinda, P.E.
Old Dominion University, 2009
Director: Dr. Duc T. Nguyen

A technique for the optimization of stability-constrained geometrically nonlinear shallow trusses with snap-through behavior is demonstrated using the arc length method and a strain energy density approach within a discrete finite element formulation. The optimization method uses an iterative scheme that evaluates the performance of the design variables and then updates them according to a recursive formula that is controlled by the arc length method. A minimum weight design is achieved when a uniform nonlinear strain energy density is found in all members. This minimal condition places the design load just below the critical-limit load that causes snap-through of the structure. The optimization scheme is programmed into a nonlinear finite element algorithm to find the large strain energy at critical-limit loads. Examples of highly nonlinear trusses that are found in the literature are presented to verify the method.

This dissertation is dedicated to

my parents,

my wife, Liz,

and my children: Amy, Rachael, Joshua, Hannah and Zachary.

ACKNOWLEDGMENTS

I would like to thank my advisor (Professor Duc Nguyen) and committee members (Professor Gene Hou and Professor Chuh Mei) for their guidance and encouragement. Special thanks to Dr. Jay Warren for helping me to appreciate more about the arc length method and also to my NASA supervisor, Dr. John Korte for his insight and suggestions. Special thanks to Dr. Crisfield, since I was fortunate to find your two valuable volumes on the arc length method. Finally, I would like to dedicate my dissertation to my family for their patience, love and encouragement during the journey.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Large aerospace truss systems have been proposed over the years to support the future of orbital space technologies. For example, since the 1960's, various research organizations have supported the deployment of space-based solar power. The U.S. Department of Defense is currently studying how to deploy a solar power constellation to supply power to remote bases or on the battlefield. Also during the 1960's, Rockwell International began to study solar power satellites (SPS) that used large space truss structures (figure 1) for supporting microwave antennae and mirrors [1]. The International Space Station currently makes use of deployable trusses (figure 2) to support its solar array panels. Flexible structures that can be stowed in very limited volumes for efficient packing into launch payloads are desirable in order to reduce mission costs. The quick deployment of these structures reduces risks by minimizing extravehicular activities while increasing reliability by having few joints. All of these space-based concepts rely on truss systems to support their mission goals because a truss is easier to assemble in microgravity and offers a minimum weight design. The lighter and more efficient these structural systems become (i.e. with very slender members), the more likely they are to have nonlinear stability problems. Civil engineering structures are also prone to instability such as shallow domes, large spanning bridges and transmission towers. The analysis challenge is to find effective techniques that go beyond the

---

The Journal of Finite Elements in Analysis and Design is used as a format model for the references cited.

identification of Euler-type bifurcation points and find the optimal truss to carry applied limit loads without becoming unstable and releasing stored strain energy.

Space trusses may suddenly collapse if their failure modes are not understood. These structures may be designed to maintain their integrity even after an initial inelastic failure. Some structures may collapse after a localized failure of one of their members. This is a progressive failure that redistributes loads causing other members to fail until the total structure becomes unstable [2]. To accurately model a progressive failure, an analysis method that can create a load vs. displacement response during instability is required. Methods that have been used involved creating finite element models that update their stiffness according to nonlinear geometric response [2].

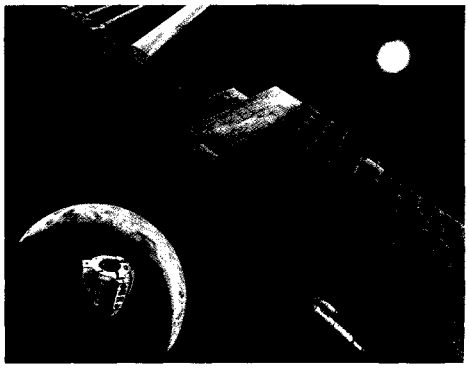


Figure 1. Solar power satellite platform.

Figure 2. International Space Station.

Another current use for optimizing snap-through type structures is in the miniaturization of electronic devices known as "micro electro mechanical systems" (MEMS) [3]. These are micro formed structures that are integrated into silicon chips. The micro structures may be formed by beams or plates that snap-through under electrostatic forces. Figure 3 shows a MEMS switch before and after snap-through. The

micro switch is a silicon beam about 25 micro meters in length that is activated by an electrostatic charge. The electronic industry uses these micro structures as switches

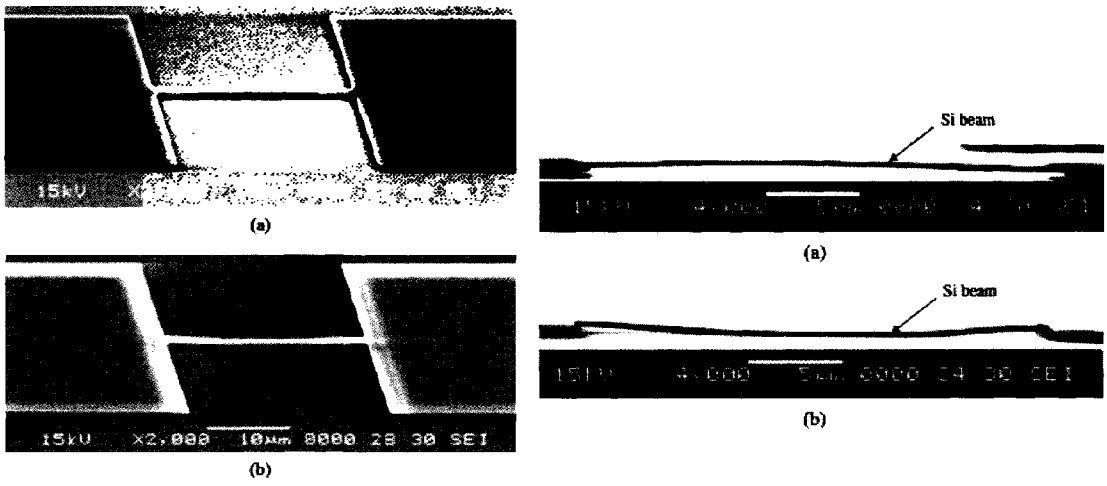


Figure 3. Photographs of fabricated MEMS switch, (a) before snap-through, (b) after snap-through [4].

which have the benefit of low power consumption, negligible heat gain and integration into micro chips [5]. Knowing the member size and electro-static load required to snap-through a micro-structure is necessary in designing the switch.

1.1 Overview

Weight optimization of slender truss members with stability constraints has been investigated by other researchers using various analytical techniques. Prager [6] discusses a general method of optimality that is subject to a single design constraint such as displacement, critical buckling load or natural frequency. Prager and Taylor [7] investigated columns under linear buckling constraints. Other investigators followed with schemes to optimize columns with multiple eigenvalues and design sensitivities with

constraints on buckling. Some of the earliest finite element optimization investigations began in the 1970's with linear buckling that ignored any large deformations. Khot, Venkayya, and Berke [8] presented an optimality criterion for minimizing weight to structures with stability requirements by using a linear eigenvalue buckling solution within a finite element analysis. Several other published works have studied the nonlinear buckling of shallow trusses. Kamat and Ruangsilasingha [9] addressed the problem of maximizing the critical load of shallow, three-dimensional (3-D) trusses. They developed sensitivity derivatives of the critical load with respect to design variables. Many case studies used in previously published works were taken from Crisfield [10] and [11]. The case examples he used challenged many nonlinear solvers to correctly follow a nonlinear snap-through load path. Crisfield used these problems to test nonlinear behavior in truss structures and to demonstrate the importance of identifying the critical-limit load along the equilibrium path.

Imperfections in the truss geometry also can reduce the load-carrying capability of a system. An investigation of random initial imperfections for frame-type structures that exhibit instability at limit points was performed by Warren [12]. He used an arc length approach to trace the nonlinear equilibrium paths of simple benchmark examples. Khot and Kamat [13] explained that the reduction in load capacity results primarily from the nonlinear response of a system and secondarily from imperfections in the geometry.

Optimizing the weight of truss structures that are undergoing large displacements requires analysis methods that accurately trace the equilibrium path and identify critical limit points. Most "shallow type" space truss designs will have snap-through response curves that are similar to those shown in figure 4. Depending on the geometry of the

design, the structure may have a snap-back path or may respond with bifurcation buckling.

The nonlinear buckling nature of "shallow-type" structures, as opposed to bifurcation or eigenvalue buckling, includes a post-buckling instability region along the equilibrium path as shown in figure 4. The nonlinear instability region is where "snap-through" occurs and the equilibrium path goes from one stable point (1) to another new stable point (2) as shown in figure 4. The nonlinear behavior places the critical limit load at 1 equal to 2 but corresponding to a new structural shape. The second stable point along the equilibrium path occurs after a large displacement of the structure. During this "snap-through," the slope of the equilibrium path (load vs. deflection) will eventually become zero. The slope of this curve is also referred to as the tangent stiffness [14]. When the tangent stiffness softens and approaches zero for a single-degree-of-freedom (sdof) system, many nonlinear solvers will encounter convergence problems. Some solvers will immediately jump to point 2 without identifying the unstable snap-through path. Bifurcation buckling is also shown in figure 4 with a linear pre-buckling region along the equilibrium path up to a critical load point (P critical). At the bifurcation point the structure immediately becomes unstable and buckles. The member is unable to support any further load which is not the case for nonlinear "snap-through" buckling. The focus of this work is on the fundamental path and not bifurcation points. This work also ignores inertial forces during snap-through and confines its interest to applying static methods.

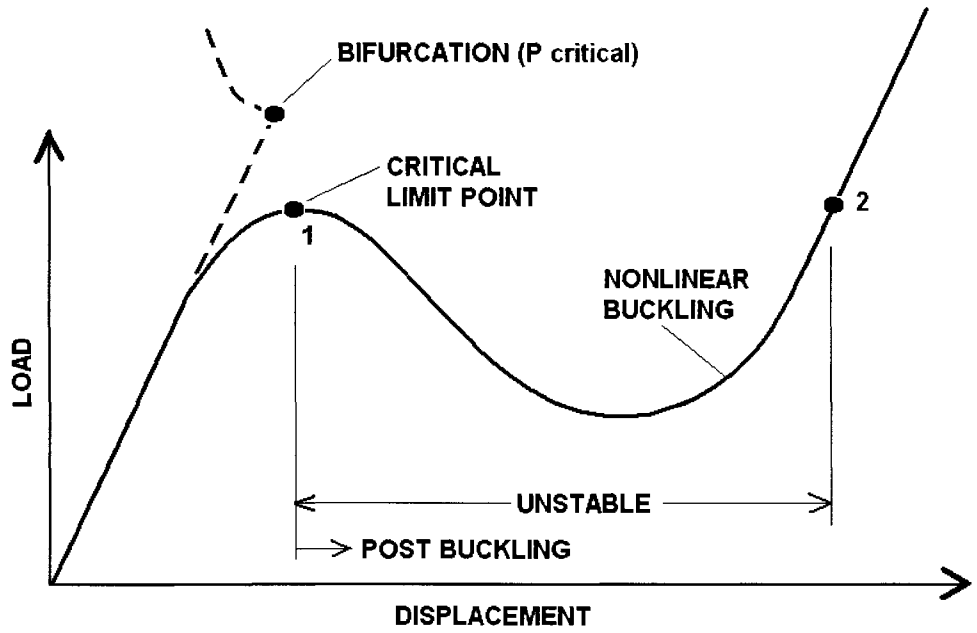


Figure 4. Equilibrium paths for nonlinear and bifurcation/eigenvalue buckling [15].

Nonlinear buckling responses result from the level of nonlinearity of the geometry in the design and the applied loading. Verification problems from Crisfield [10], [11] and [16] show the snap-through and snap-back behavior of several shallow truss designs. Achieving a minimum weight design of shallow trusses similar to the design examples from Crisfield requires limit-point stability constraints that can maximize critical load.

In [17], Khot, Kamat, and Venkayya offer exact closed-form solutions to simple shallow truss problems that have been optimized for weight. They make use of the minimization of the total potential energy that maximizes the load-carrying capability without instability. Optimizations of geometrically nonlinear trusses that are similar to those used by Crisfield were attempted later by Khot and Kamat [13]. These examples highlight the challenges that have been encountered with many analysis techniques in

following a nonlinear response path up to a critical-limit load. The sample problems also fully expose the short-comings of modern finite element programs by their lack of optimization scheme for stability- constrained problems. Sedaghati et al. [18] developed an optimization technique using the group theoretic approach. The method has stability constraints that use a strain energy density recurrence relationship to update the design variables. The technique involves obtaining a uniform strain energy density in all members to establish a minimum weight design. Sedaghati et al. further made use of shallow flexible truss structures to test and verify their methods.

The proposed Hrinda method was first used to investigate stability of shallow trusses prone to snap-through. These structures lose their stability differently than structures undergoing bifurcation buckling. Finite element programs typically solve for elastic stability by running an eigenvalue buckling solution. However, shallow truss structures will typically lose their ability to support any loads well below their bifurcation load. If a standard eigenvalue buckling solution is run in NASTRAN for a shallow truss structure, the load predicted in the analysis may be much higher than the snap-through load predicted using the arc length method in the proposed Hrinda program. Also, the use of the arc length method within the Hrinda program overcomes the limitations of classical solution techniques such as the Newton-Raphson method.

The solutions described by Khot, Kamat and Venkayya in [13], [17] and by Sedaghati et al. [18] rely on their own algorithms. These references do not elaborate on the nonlinear solver used by the authors. Also, the references do not mention combining nonlinear solver parameters to an optimization scheme as Hrinda has proposed. The Hrinda method also solves for the optimal weight within a finite element formulation

using advanced sparse techniques developed by Nguyen in [19]. The above references do not discuss any use of computer resources in their nonlinear solutions.

The NASTRAN nonlinear finite element analysis had convergence difficulties tracing the equilibrium paths of the examples in chapter 4. Careful selection of parameters used in NASTRAN eventually produced results found by Khot, Kamat and Venkayya in [13], [17] and by Sedaghati et al. [18].

1.2 Objective

The objective of this work is to develop an optimal design by using the principal of strain energy density and parameters in the arc length method to trace the nonlinear equilibrium path up to an instability point. A stable structural system exists when deformations are increasing as the applied load is increased; an unstable system exists when deformations are still increasing but the loads are decreasing. The Hrinda optimization method used in this work achieves an optimal design by combining the arc length method with a design-variable update scheme that identifies a uniform nonlinear strain energy density in all members. The method uses an iterative scheme that evaluates the performance of the design variables and then updates them according to a recursive formula that is controlled by the arc length method. The method is programmed into a new finite element algorithm developed from [20]. The Hrinda optimization algorithm is used to test the arc length updating scheme. The finite element program contains a 3-D truss element and an arc length method that could solve the snap-through and snap-back response of structures presented by Crisfield in [10], [11] and [16]. Examples of highly nonlinear trusses that have been found in the literature are presented to verify the method.

Only conservative systems that adhere to the theorem of minimum potential energy are considered. These are stable systems that are subjected to quasi-static proportional loads, where the equations of equilibrium make the potential energy stationary under small displacements [15].

CHAPTER II

ARC LENGTH METHOD

The Riks-Wempner arc length method is used in this work to solve highly geometric nonlinear structures that are to be weight optimized. Various versions of the arc length method have been presented by many researchers [10], [11], [12], [21] and [22] and incorporated into the finite element analysis procedures. For simplicity and to facilitate the discussion in subsequent sections, the Riks-Wempner arc length method is reviewed in section 2.1. The Riks-Wempner method is considered a partial arc length method because it relies on a normal to the tangent rather than the circular path to search for an equilibrium point [21]. The details of the iterative process along the normal are shown in figure 7 along with accompanying equations. The nonlinear equilibrium path is incrementally produced with load steps and convergence tests as depicted in figure 7.

2.1 Review of Arc Length Formulas

The Riks-Wempner arc length method traces the nonlinear equilibrium path by using an iterative process that begins by computing initial displacements Δq_0 based on a user-defined load increment $\Delta\lambda_0$, as shown in figure 5. The linear stiffness K_{T_0} is used to start the process and is replaced with the tangent stiffness K_{T_i} in further iterations.

The initial displacements Δq_0 are found using

$$\frac{\Delta\lambda_0}{\Delta q_0} = \frac{\lambda}{\Delta q_{\text{tot}}} \tag{2.1}$$

where $\lambda = 1$ and Δq_{tot} is derived from the expression

$$K_{T_0}\Delta q_{\text{tot}} = \lambda Q \tag{2.2}$$

where Q is the total applied loading. The notations that are given here for the displacements and loads are vector quantities with the stiffness notation understood as a matrix. Substituting into equation (2.1) yields

$$\frac{\Delta\lambda_0}{\Delta q_0} = \frac{\lambda = 1}{(\Delta q_{\text{tot}})_0} \tag{2.3}$$

which is solved for the initial displacement of

$$\Delta q_0 = \Delta\lambda_0 (\Delta q_{\text{tot}})_0 \tag{2.4}$$

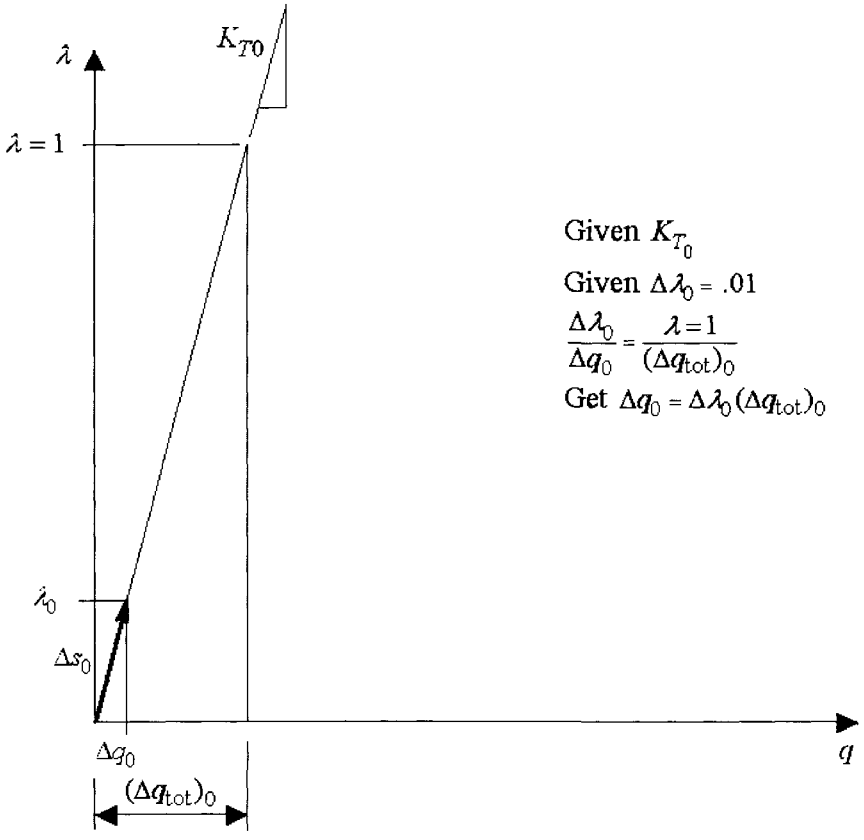


Figure 5. Starting the arc length.

The method then finds the next equilibrium point from the initial point 0 by using the tangent stiffness K_{T_1} as shown in figure 6. The tangent stiffness matrix is assembled by

using the nonlinear truss that is given in equation (3.30) in chapter 3; it is defined as the elastic linear stiffness that is given by equation (3.14) and added to the nonlinear geometric stiffness in equation (3.18). The sum of the linear elastic and nonlinear geometric matrices produces the global tangent stiffness at point *i* along the load-displacement path of the single-degree-of-freedom system. A new displacement Δq_{tot_i} is calculated from

$$K_{T_i} \Delta q_{\text{tot}_i} = \lambda Q \tag{2.5}$$

with $\lambda = 1$ as before.

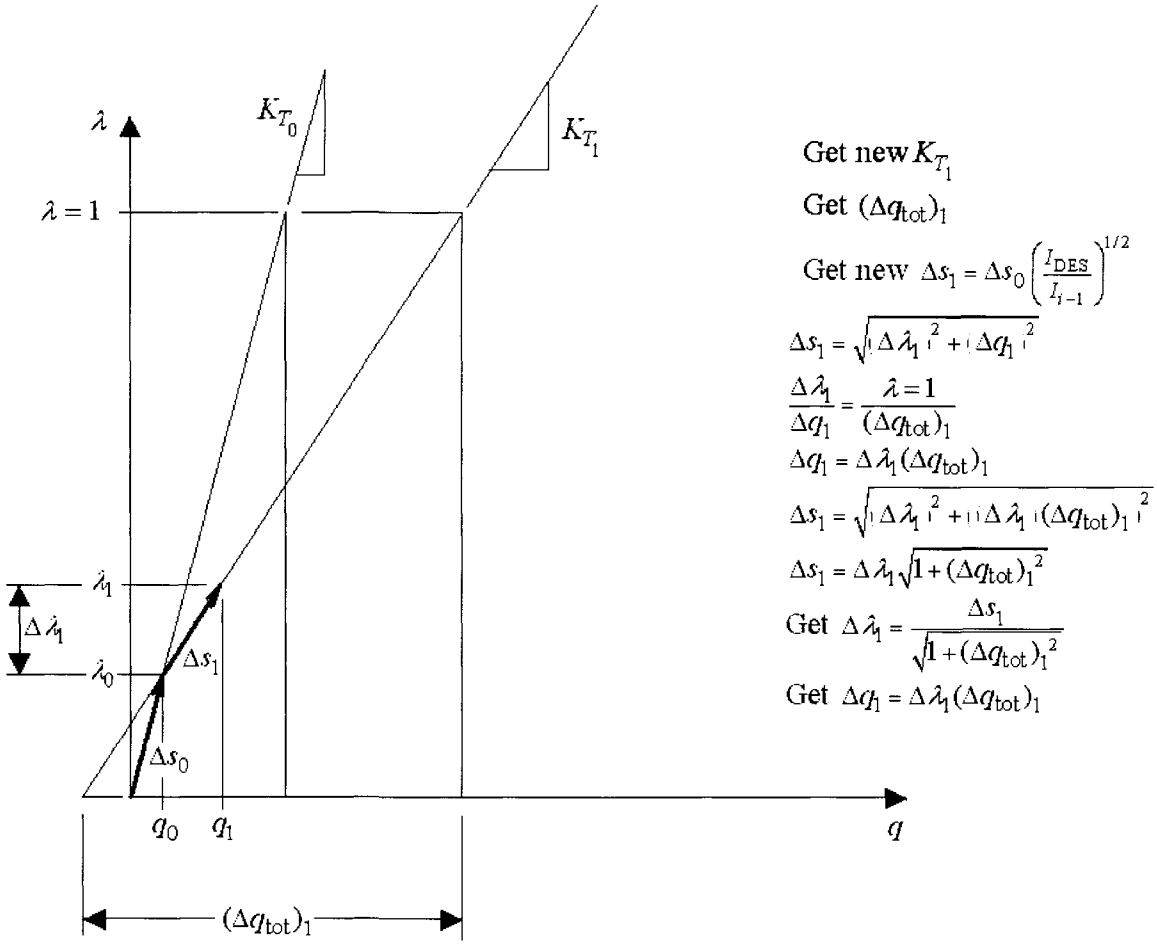


Figure 6. Next iteration.

The arc length used in the Riks-Wempner method is the straight line Δs_i ; this arc length is either constant or scaled by the user input with the following

$$\Delta s_i = \Delta s_{i-1}\left(\frac{I_{\text{DES}}}{I_{i-1}}\right)^{\frac{1}{2}} \tag{2.6}$$

The user decides on the required number of iterations I_{i-1} and on the number of desired iterations I_{des}. The arc length Δs_i is the tangent vector along the equilibrium path and can be calculated as

$$\Delta s_0 = \sqrt{\Delta\lambda_0^2 + (\Delta q^T)_0 \Delta q} \tag{2.7}$$

or by using

$$\Delta s_0 = \Delta\lambda_0\sqrt{1 + \Delta q_{\text{tot}}^T \Delta q_{\text{tot}}} \tag{2.8}$$

The length of the next tangent vector Δs_1 can now be written as

$$\Delta s_1 = \Delta\lambda_1\sqrt{1 + (\Delta q_{\text{tot}})_1^2} \tag{2.9}$$

In this equation, the unknown is $\Delta\lambda_1$, which can be solved by using

$$\Delta\lambda_1 = \frac{\Delta s_1}{\sqrt{1 + (\Delta q_{\text{tot}})_1^2}} \tag{2.10}$$

the incremental displacement can be found by using

$$\Delta q_1 = \Delta\lambda_1 (\Delta q_{\text{tot}})_1 \tag{2.11}$$

The next step is a test for convergence that first obtains the internal force F_1. The residual forces can be calculated as

$$F_{R_1} = \lambda_1 Q - F_1 \tag{2.12}$$

Here, the residual forces are the differences in the internal forces that are calculated from the equilibrium path and are shown in figure 7.

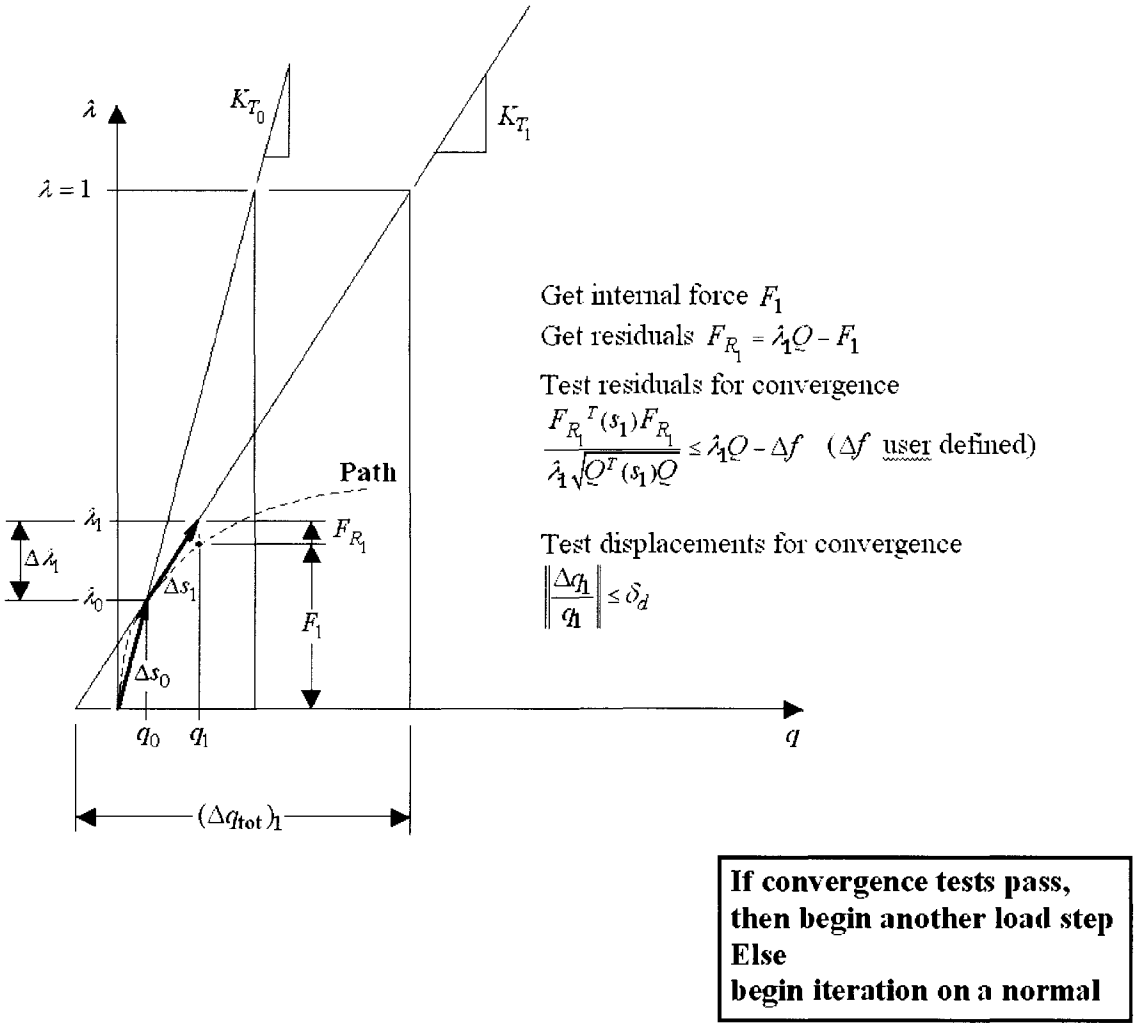


Figure 7. Convergence tests.

The following is used to test for load convergence:

$$\frac{F_{R_1}{}^T(s_1)F_{R_1}}{\lambda_1\sqrt{Q^T(s_1)Q}} \le \lambda_1 Q - \Delta f \tag{2.13}$$

where Δf is a user-defined tolerance value. If the statement is true, then the loads test is successful.

A convergence for displacements is also performed by using the ratio

$$\left\| \frac{\Delta q_1}{q_1} \right\| \leq \delta_d \tag{2.14}$$

where δ_d is also a user-defined tolerance. Once again, if the statement is true, then the displacement test is passed.

If both convergence tests are passed, then another load step is started by following the same procedure outlined above. If, however, one of the tests fails, iteration on the normal begins as shown in figure 8. The objective is to find q_2 and λ_2 by iterating down the normal from the tangent vector Δs_1. Figure 8 shows the necessary geometry that is required to begin the iteration. A portion of the figure is magnified to show how the variables are defined.

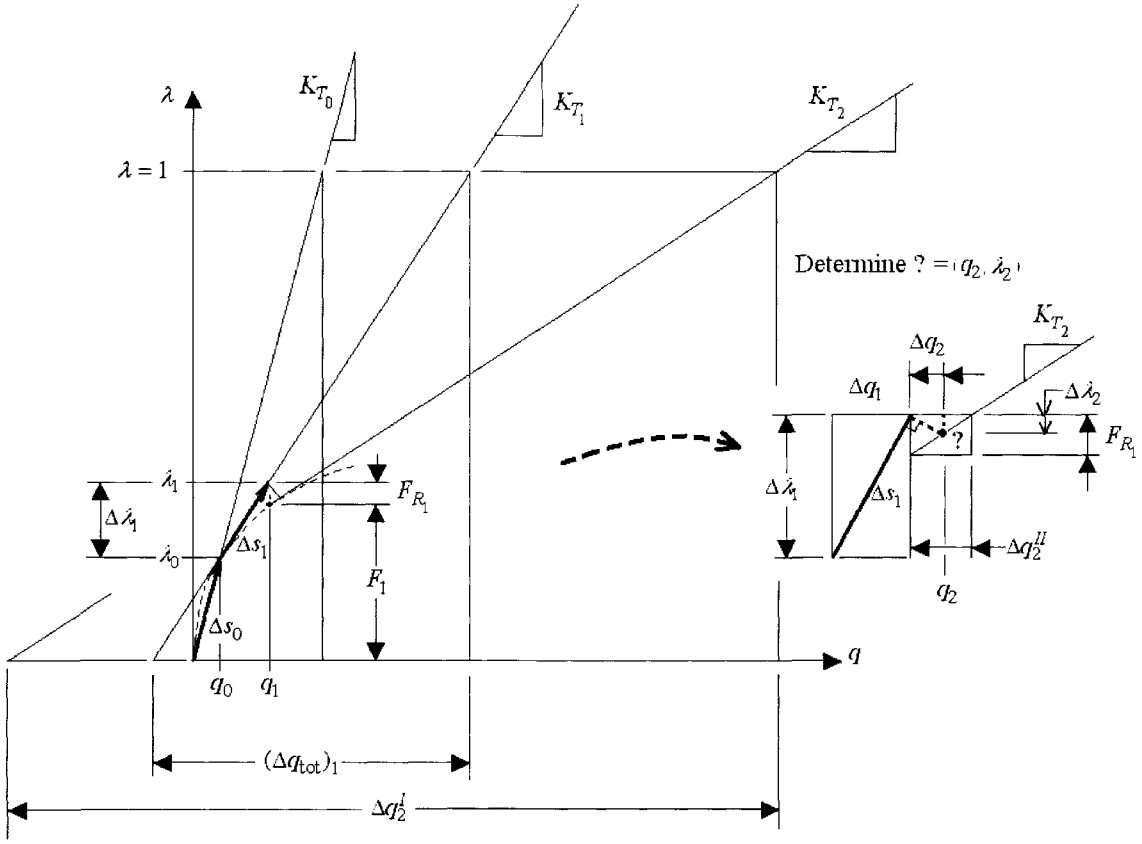


Figure 8. Iteration on a normal.

The vector n_2 is normal to Δs_1 if

$$\frac{\Delta q_1}{\Delta \lambda_1} = \frac{\Delta \lambda_2}{\Delta q_2} \tag{2.15}$$

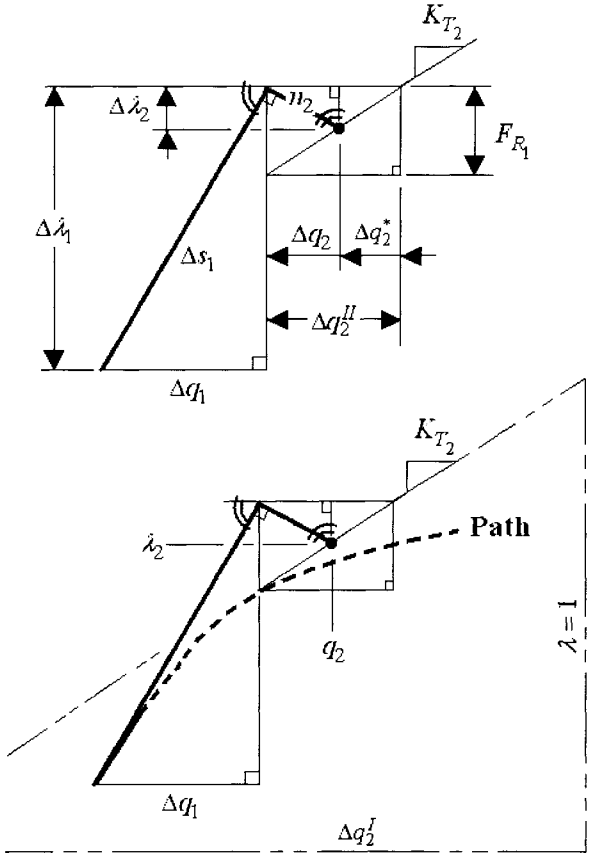
The equation comes from using the method of similar triangles as shown in figure 9. The relationship may also be written as

$$\Delta q_1 \Delta q_2 - \Delta \lambda_1 \Delta \lambda_2 = 0 \tag{2.16}$$

A new tangent stiffness $\left[K_{T_2}\right]$ is calculated and used in the following:

$$\left[K_{T_2}\right] \Delta q_2^{II} = F_{R_1} \tag{2.17}$$

where all variables are defined in figure 9.



If n_2 is $\perp \Delta s_1$

Use similar triangles:

$$\frac{\Delta q_1}{\Delta \lambda_1} = \frac{\Delta \lambda_2}{\Delta q_2}$$

$$\Delta q_1 \Delta q_2 - \Delta \lambda_1 \Delta \lambda_2 = 0$$

Get $\left[K_{T_2}\right]$

$$\left[K_{T_2}\right] \Delta q_2^{II} = F_{R_1}$$

$$\Delta q_2^{II} = \left[K_{T_2}\right]^{-1} F_{R_1}$$

$$\Delta q_2 = \Delta q_2^{II} - \Delta q_2^*$$

$$\frac{\Delta \lambda_2}{\Delta q_2^*} = \frac{(\lambda = 1)}{\Delta q_2^I}$$

$$\Delta q_2^* = \frac{\Delta \lambda_2 \Delta q_2^I}{(\lambda = 1)} = \Delta \lambda_2 \Delta q_2^I$$

$$\Delta q_2 = \Delta q_2^{II} - \Delta \lambda_2 \Delta q_2^I$$

$$\Delta q_1 \left(\Delta q_2^{II} - \Delta \lambda_2 \Delta q_2^I \right) - \Delta \lambda_1 \Delta \lambda_2 = 0$$

$$\Delta q_1 \Delta q_2^{II} - \Delta q_1 \left(\Delta \lambda_2 \Delta q_2^I \right) - \Delta \lambda_1 \Delta \lambda_2 = 0$$

$$\Delta q_1 \Delta q_2^{II} - \Delta \lambda_2 \left(\Delta q_1 \Delta q_2^I + \Delta \lambda_1 \right) = 0$$

$$\Delta \lambda_2 = \frac{\Delta q_1 \Delta q_2^{II}}{\left(\Delta q_1 \Delta q_2^I + \Delta \lambda_1 \right)}$$

$$\lambda_2 = \lambda_0 + \Delta \lambda_1 + \Delta \lambda_2$$
$$q_2 = q_0 + \Delta q_1 + \Delta q_2$$

Figure 9. Details of iteration on a normal.

Equation (2.17) can be rewritten as

$$\Delta q_2^{II} = \left[K_{T_2} \right]^{-1} F_{R_1} \tag{2.18}$$

The incremental displacement shown in figure 9 can be expressed as

$$\Delta q_2 = \Delta q_2^{II} - \Delta q_2^{*} \tag{2.19}$$

Using similar triangles, the following can now be obtained

$$\frac{\Delta \lambda_2}{\Delta q_2^{*}} = \frac{(\lambda = 1)}{\Delta q_2^{I}} \tag{2.20}$$

Terms can be rearranged to obtain

$$\Delta q_2^{*} = \frac{\Delta \lambda_2 \Delta q_2^{I}}{(\lambda = 1)} = \Delta \lambda_2 \Delta q_2^{I} \tag{2.21}$$

Combining equations (2.19) and (2.21) creates

$$\Delta q_2 = \Delta q_2^{II} - \Delta \lambda_2 \Delta q_2^{I} \tag{2.22}$$

Now, equation (2.16) is combined with equation (2.22) to obtain

$$\Delta q_1 \left(\Delta q_2^{II} - \Delta \lambda_2 \Delta q_2^{I} \right) - \Delta \lambda_1 \Delta \lambda_2 = 0$$

which can be reduced to

$$\Delta \lambda_2 = \frac{\Delta q_1 \Delta q_2^{II}}{\left(\Delta q_1 \Delta q_2^{I} + \Delta \lambda_1 \right)} \tag{2.23}$$

and added to the previous load increments as follows:

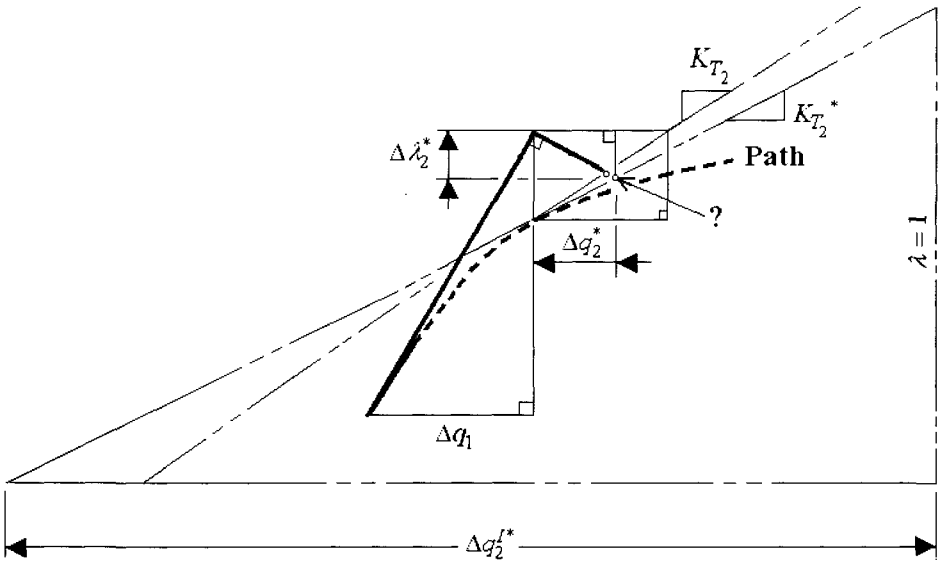
$$\lambda_2 = \lambda_0 + \Delta \lambda_1 + \Delta \lambda_2 \tag{2.24}$$

Likewise, the displacement increments are summed by using equation (2.22) to yield

$$q_2 = q_0 + \Delta q_1 + \Delta q_2 \tag{2.25}$$

Equations (2.24) and (2.25) represent the load and displacement at location 2 along the equilibrium path. A convergence check is performed similar to that shown in figure 7 and in equations (2.12) through (2.14). If the test is passed, then a new load increment begins by using the tangent vector of equation (2.9). If the tests fail, then iteration down the normal vector n is started as shown in figure 10. The process continues until the convergence tests for load and displacement are both passed.

Structured state-of-the-art sparse equation solver technology such as sparse assembly algorithms, reordering algorithms to minimize fill-in-terms, and super node strategies for efficient unrolling techniques during numerical factorization phase were all incorporated into equations (2.5) and (2.18) of the arc length procedure [19] .



Get $\left[K_{T_2}^*\right]$

Get Δq_2^* and $\Delta\lambda_2^*$ using previous scheme

Updated displacements and load:

$\lambda_2 = \lambda_0 + \Delta\lambda_2^*$

$q_2 = q_0 + \Delta q_2^*$

Figure 10. Continuing iteration on normal.

2.2 Linear/Nonlinear, Symmetrical/Unsymmetrical Equation Tests

The following equations were used in the proposed algorithm to test the arc length method and computer coding. Their simple mathematical solutions were a quick check of the arc length solver portion of the algorithm without utilizing the finite element sparse solver. The first system of equations are shown in equations (2.26) and (2.27). The equations are written in matrix format (2.28) to allow for input to the proposed algorithm.

$$24u_1^2 - 12u_2 = 0 \tag{2.26}$$

$$-12u_1 + 8u_2^2 = 20 \tag{2.27}$$

$$\begin{pmatrix} 24u_1 & -12 \\ -12 & 8u_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 20 \end{pmatrix} \tag{2.28}$$

The unknowns, (u_1, u_2), in (2.28) have a unique solution that is (-.7633, 1.165).

The next equation pair is shown below in (2.29) and (2.30). The equations are also placed into matrix form in (2.31).

$$24u_1 - 12u_2 = -2.0 \tag{2.29}$$

$$-12u_1 + 8u_2 = 0.0 \tag{2.30}$$

$$\begin{pmatrix} 24 & -12 \\ -12 & 8 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -2.0 \\ 0.0 \end{pmatrix} \tag{2.31}$$

The solution to the unknowns (u_1, u_2) is given as (-.3333,-.5000) respectively.

Another equation test used the system in (2.32) and (2.33). As performed in the previous tests, the equations are arranged in (2.34) for proper computer input format.

$$24u_1^3 - 16u_2 = 8 \tag{2.32}$$

$$-36u_1 + 6u_2^2 = -24 \tag{2.33}$$

$$\begin{pmatrix} 24u_1^2 & -16 \\ -36 & 6u_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}=\begin{pmatrix} 8 \\ -24 \end{pmatrix} \tag{2.34}$$

The solution to the two unknowns is (.6675, -.05417).

Another equation pair is shown in (2.38) and (2.39) with the matrix form given in (2.40).

$$10u_1^2 + 2u_2^2 = 0.0 \tag{2.38}$$

$$2u_1^2 + 2u_2^2 = 0.0 \tag{2.39}$$

$$\begin{pmatrix} 10u_1 & 2u_2 \\ 2u_1 & 2u_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}=\begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \tag{2.40}$$

The solution to the two unknowns (u_1, u_2) in (2.40) is (0,0).

The equations in this section were solved with the arc length method in the proposed algorithm. The solutions shown in this section were all verified by the algorithm. Further testing was conducted using the nonlinear truss element as discussed in the next chapter.

A listing of the computer input files is given in appendices A.1-A.4.

CHAPTER III

GEOMETRICALLY NONLINEAR 3-D TRUSS ELEMENT STIFFNESS

In this work, we are interested in the behavior of 3-D truss members undergoing large displacements. In order to use and test the Hrinda optimization scheme, it became necessary to verify a 3-D truss stiffness matrix that could behave properly while undergoing large deflections. This requires a modification to the elastic stiffness that is typically used in the linear equation. The development of a properly formed geometric nonlinear stiffness matrix begins by altering the basic linear equation [23] of the applied force $\vec{F}$ with the stiffness $[K]$ and nodal displacements $\vec{u}$, which can be written as

$$\vec{F} = [K]\vec{u} \tag{3.1}$$

The relationship is modified by introducing a nonlinear geometric matrix $[K]_G$ that is added to the elastic stiffness $[K]_E$ and written as

$$[K]_T = [K]_E + [K]_G \tag{3.2}$$

where $[K]_T$ is the tangent stiffness. The following sections offer a detailed formulation of the tangent stiffness matrix given by equation (3.2) and the testing that was required to assure it was working properly.

3.1 Derivation of 3-D Geometrically Nonlinear Truss Element Stiffness

Trusses that are undergoing large deflections must be analyzed for the deformed geometry of the structure. The linear equation (3.1) that relates the applied force $\vec{F}$ with

the truss element stiffness $[K]$ and displacements $\vec{u}$ must be modified to account for changes in nodal geometry as the load is applied. The standard elastic stiffness matrix $[K]_E$ is modified by the addition of a geometric stiffness matrix $[K]_G$ that depends on the geometry and the initial internal forces.

Note that efficient sparse assembly procedures [19] have been used in equation (3.2), while efficient sparse reordering algorithms (i.e., both to minimize fill-in terms and during the factorization phase) and sparse solvers with unrolling strategies [19] have been utilized in equation (3.2). The sparse assembly techniques are discussed in chapter 5.

Nonlinear analysis is usually divided into a sequence of load steps where the displacements are being solved after each load increment. The nonlinear stiffness is then updated with the displacement and the internal force values from the previous step. A discussion follows of the nonlinear truss element that is used in the finite element program in this study.

The following definitions are used to derive the elastic truss stiffness from the variables that are defined in figure 11. The element's original length is found by using the global nodal coordinates given as x_1, y_1, z_1 at end 1 and x_2, y_2, z_2 for end 2 to give

$$l_0 = \sqrt{(x_{21}^{\ 2} + y_{21}^{\ 2} + z_{21}^{\ 2})} \tag{3.3}$$

where

$$x_{21} = (x_2 - x_1) \tag{3.4}$$

$$y_{21} = (y_2 - y_1) \tag{3.5}$$

$$z_{21} = (z_2 - z_1) \tag{3.6}$$

The change in displacements for an *i*th truss can be written as

$$x_{21D} = x_2 - x_1 + ux_2 - ux_1 \tag{3.7}$$

$$y_{21D} = y_2 - y_1 + uy_2 - uy_1 \tag{3.8}$$

$$z_{21D} = z_2 - z_1 + uz_2 - uz_1 \tag{3.9}$$

where ux_1, uy_1, uz_1 are the displacements at end 1 and ux_2, uy_2, uz_2 are the displacements at end 2. This may be used to write the final length of the element as

$$li = \sqrt{(x_{21D}{}^2 + y_{21D}{}^2 + z_{21D}{}^2)} \tag{3.10}$$

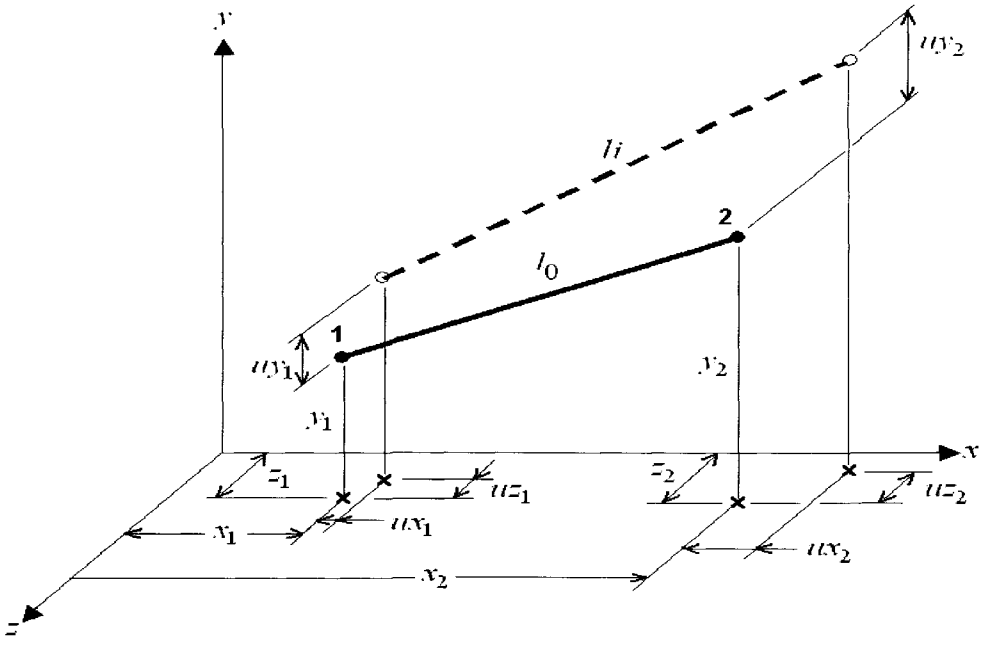


Figure 11. Nonlinear 3-D truss variables.

The tangent stiffness $[K]_T$ given by equation (3.2) is can be defined as outlined by Crisfield in [10] as

$$[K]_T = \frac{\partial \vec{q}_i}{\partial \vec{p}} \tag{3.11}$$

where $\frac{\partial \vec{q}_i}{\partial \vec{p}}$ is the rate of change of the internal forces, $\vec{q}_i$, as the nodal displacements, $\vec{p}$, are changing. The tangent stiffness matrix is expanded by Crisfield into

$$[K]_T = \frac{\partial \vec{q}_i}{\partial \vec{p}} = 2\alpha A_o \vec{b} \frac{\partial \sigma}{\partial \vec{p}} + 2\alpha A_o \frac{\partial \vec{b}}{\partial \vec{p}} \sigma \tag{3.12}$$

The left side of the equation, $2\alpha A_o \vec{b} \frac{\partial \sigma}{\partial \vec{p}}$, is the standard elastic stiffness matrix for a 3D truss, $[K]_E$. The term to the right of the addition sign, $2\alpha A_o \frac{\partial \vec{b}}{\partial \vec{p}} \sigma$, is the nonlinear geometric stiffness matrix, $[K]_G$. In defining equation (3.12) the following definitions are used: A_o =cross sectional area, σ is the stress, 2α =original length, l_0.

The elastic stiffness matrix is derived as

$$2\alpha A_o \vec{b} \frac{\partial \sigma}{\partial \vec{p}} = \frac{A_o E}{8\alpha^3} \begin{bmatrix} -x_{21D} \\ x_{21D} \\ -y_{21D} \\ y_{21D} \\ -z_{21D} \\ z_{21D} \end{bmatrix} \begin{bmatrix} -x_{21D} & x_{21D} & -y_{21D} & y_{21D} & -z_{21D} & z_{21D} \end{bmatrix} \tag{3.13}$$

This may be expanded using $8\alpha^3 = l_0^3$ to give the standard elastic stiffness

$$[K]_E = \frac{A_o E}{l_0^3}\begin{bmatrix} x_{21D}^2 & -x_{21D}x_{21D} & x_{21D}y_{21D} & -x_{21D}y_{21D} & x_{21D}z_{21D} & -x_{21D}z_{21D} \\ & x_{21D}^2 & -x_{21D}y_{21D} & x_{21D}y_{21D} & -x_{21D}z_{21D} & x_{21D}z_{21D} \\ & & y_{21D}^2 & -y_{21D}y_{21D} & y_{21D}z_{21D} & -y_{21D}z_{21D} \\ & & & y_{21D}^2 & -y_{21D}z_{21D} & y_{21D}z_{21D} \\ & SYM & & & z_{21D}^2 & -z_{21D}z_{21D} \\ & & & & & z_{21D}^2 \end{bmatrix} \tag{3.14}$$

The geometric stiffness matrix is found by using the right hand side of equation (3.12)

$$2\alpha A_o \frac{\partial \vec{b}}{\partial \vec{p}}\sigma = \frac{A_o \sigma}{2\alpha}[B] \tag{3.15}$$

The strain-displacement vector in the above equation is obtained by taking the derivative of the Green's strain vector with respect to the nodal displacement vector to give

$$\frac{\partial \vec{b}}{\partial \vec{p}} = \frac{1}{l_0^2}\begin{bmatrix} -u_{21} \\ u_{21} \\ -u_{21} \\ u_{21} \\ -u_{21} \\ u_{21} \end{bmatrix} \tag{3.16}$$

This type of strain measurement will be explained later in this chapter.

Also used in equation (3.15) is

$$[B] = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 \\ & & 1 & -1 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & SYM & & & 1 & -1 \\ & & & & & 1 \end{bmatrix} \tag{3.17}$$

The geometric stiffness may be written using equation (3.15), (3.16) and $\sigma = \frac{AN}{A_0}$

$$[K]_G = \frac{AN}{l_0}\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 \\ & & 1 & -1 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & SYM & & & 1 & -1 \\ & & & & & 1 \end{bmatrix} \tag{3.18}$$

with AN equal to the internal elemental force. Equation (3.18) is further defined in the next section by defining the internal forces. Section 3.3 will then use this information to develop the full tangent stiffness.

3.2 Element Internal Force

The internal forces in the truss element are required for the geometric stiffness and are defined by using matrix notation. The element strain formulation uses a constant cross-sectional area and assumes that the length and area of the truss will remain large. The strain energy or the work done is one-half of the nodal forces multiplied by the corresponding deflections.

The internal force in the truss is now defined to include nonlinear effects. The force will be axial and is needed to update the nonlinear stiffness matrix. Defining the force in matrix notation is necessary for efficient computer programming. The internal force may be defined as

$$AN = EA_0\varepsilon \tag{3.19}$$

where Green's strain as formulated in chapter 6 is

$$\varepsilon = \frac{li^2 - l_0^2}{2l_0^2} \tag{3.20}$$

Substituting equations (3.3) and (3.10) into the strain equation leads to

$$\varepsilon = \frac{\left[\left[(x_2 - x_1) \quad (y_2 - y_1) \quad (z_2 - z_1)\right] + \left[(ux_2 - ux_1) \quad (uy_2 - uy_1) \quad (uz_2 - uz_1)\right]\right]\begin{bmatrix}(x_2 - x_1) + (ux_2 - ux_1)\\(y_2 - y_1) + (uy_2 - uy_1)\\(z_2 - z_1) + (uz_2 - uz_1)\end{bmatrix} - \left[(x_2 - x_1) \quad (y_2 - y_1) \quad (z_2 - z_1)\right]\begin{bmatrix}(x_2 - x_1)\\(y_2 - y_1)\\(z_2 - z_1)\end{bmatrix}}{2\left[(x_2 - x_1) \quad (y_2 - y_1) \quad (z_2 - z_1)\right]\begin{bmatrix}(x_2 - x_1)\\(y_2 - y_1)\\(z_2 - z_1)\end{bmatrix}} \tag{3.21}$$

This equation must be simplified to better adapt for computer programming. Therefore the following relationship is identified in the left part of the numerator

$$\left[(x_2 - x_1) \quad (y_2 - y_1) \quad (z_2 - z_1)\right] + \left[(ux_2 - ux_1) \quad (uy_2 - uy_1) \quad (uz_2 - uz_1)\right] \tag{3.22}$$

that can be written as

$$\left[(x_2 - x_1) + (ux_2 - ux_1) \quad (y_2 - y_1) + (uy_2 - uy_1) \quad (z_2 - z_1) + (uz_2 - uz_1)\right] \tag{3.23}$$

This expression may now replace the left part of equation (3.21) as

$$\varepsilon = \frac{\left[(x_2 - x_1) + (ux_2 - ux_1) \quad (y_2 - y_1) + (uy_2 - uy_1) \quad (z_2 - z_1) + (uz_2 - uz_1)\right]\begin{bmatrix}(x_2 - x_1) + (ux_2 - ux_1)\\(y_2 - y_1) + (uy_2 - uy_1)\\(z_2 - z_1) + (uz_2 - uz_1)\end{bmatrix} - \left[(x_2 - x_1) \quad (y_2 - y_1) \quad (z_2 - z_1)\right]\begin{bmatrix}(x_2 - x_1)\\(y_2 - y_1)\\(z_2 - z_1)\end{bmatrix}}{2\left[(x_2 - x_1) \quad (y_2 - y_1) \quad (z_2 - z_1)\right]\begin{bmatrix}(x_2 - x_1)\\(y_2 - y_1)\\(z_2 - z_1)\end{bmatrix}} \tag{3.24}$$

Multiplication is performed in the numerator to give

$$2(x_2 - x_1)(ux_2 - ux_1) + (ux_2 - ux_1)^2 + 2(y_2 - y_1)(uy_2 - uy_1) + (uy_2 - uy_1)^2 + 2(z_2 - z_1)(uz_2 - uz_1) + (uz_2 - uz_1)^2 \tag{3.25}$$

The denominator in equation (3.24) is just $2l_0{}^2$ which leads to

$$\varepsilon = \frac{2(x_2 - x_1)(ux_2 - ux_1) + (ux_2 - ux_1)^2 + 2(y_2 - y_1)(uy_2 - uy_1) + (uy_2 - uy_1)^2 + 2(z_2 - z_1)(uz_2 - uz_1) + (uz_2 - uz_1)^2}{2l_0{}^2} \tag{3.26}$$

where

$$2l_0{}^2 = 2\begin{bmatrix}(x_2 - x_1) & (y_2 - y_1) & (z_2 - z_1)\end{bmatrix}\begin{bmatrix}(x_2 - x_1)\\(y_2 - y_1)\\(z_2 - z_1)\end{bmatrix} \tag{3.27}$$

This simplifies equation (3.26) to

$$\frac{1}{l_0^2}\left[(x_2 - x_1)(ux_2 - ux_1) + 2(y_2 - y_1)(uy_2 - uy_1) + 2(z_2 - z_1)(uz_2 - uz_1) + \frac{1}{2}(ux_2 - ux_1)^2 + \frac{1}{2}(uy_2 - uy_1)^2 + \frac{1}{2}(uz_2 - uz_1)^2\right] \tag{3.28}$$

Substituting (3.28) into the internal force equation (3.19) and using (3.4)-(3.6) gives

$$AN = \frac{EA_0}{l_0^2}\left[x_{21}(ux_2 - ux_1) + y_{21}(uy_2 - uy_1) + z_{21}(uz_2 - uz_1) + \frac{(ux_2 - ux_1)^2}{2} + \frac{(uy_2 - uy_1)^2}{2} + \frac{(uz_2 - uz_1)^2}{2}\right] \tag{3.29}$$

3.3 Nonlinear 3-D Truss Element Stiffness

The tangent stiffness can now be written by substituting into equation (3.2) the elastic stiffness (3.14) and the geometric stiffness (3.18) with the internal force equation (3.29) to give

$$[K]_T = \frac{A_o E}{l_0^{\,3}} \begin{bmatrix} x_{21D}^{\,2} & -x_{21D}x_{21D} & x_{21D}y_{21D} & -x_{21D}y_{21D} & x_{21D}z_{21D} & -x_{21D}z_{21D} \\ & x_{21D}^{\,2} & -x_{21D}y_{21D} & x_{21D}y_{21D} & -x_{21D}z_{21D} & x_{21D}z_{21D} \\ & & y_{21D}^{\,2} & -y_{21D}y_{21D} & y_{21D}z_{21D} & -y_{21D}z_{21D} \\ & & & y_{21D}^{\,2} & -y_{21D}z_{21D} & y_{21D}z_{21D} \\ & SYM & & & z_{21D}^{\,2} & -z_{21D}z_{21D} \\ & & & & & z_{21D}^{\,2} \end{bmatrix} +$$

$$\frac{A_0 E\left[x_{21}(ux_2 - ux_1) + y_{21}(uy_2 - uy_1) + z_{21}(uz_2 - uz_1) + \frac{(ux_2 - ux_1)^2}{2} + \frac{(uy_2 - uy_1)^2}{2} + \frac{(uz_2 - uz_1)^2}{2} \right]}{l_0^{\,3}} \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 \\ & & 1 & -1 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & SYM & & & 1 & -1 \\ & & & & & 1 \end{bmatrix} \quad (3.30)$$

CHAPTER IV

NONLINEAR TRUSS VERIFICATION TESTS

The proposed finite element program was tested and verified using several examples found in the literature with nonlinear NASTRAN solutions. Of particular interest was the ability to reproduce snap-through and snap-back behavior found in some structures. The examples were chosen as a robust test of this unstable behavior and brought confidence to the computer coding and numerical techniques used in the proposed algorithm.

4.1 Single-Degree-of-Freedom Nonlinear Example

The performance of the truss element was tested by Hrinda [20] against the snap-through and snap-back problems that were identified by Crisfield [10] and [16]. The criteria for a desirable element were the ability to accurately trace an equilibrium path and the ability to reach and pass the critical-limit points within an arc length solution. In [20], finite element analysis of the Crisfield snap-through and snap-back models were preformed using the tangent stiffness given by equation 3.30 and NASTRAN. The first performance example was a single-degree-of-freedom finite element model used to demonstrate the arc length algorithm and verify results with NASTRAN and an exact solution. The problem is used in Crisfield [10] with the variables used to express the exact response equation shown in figure 12.

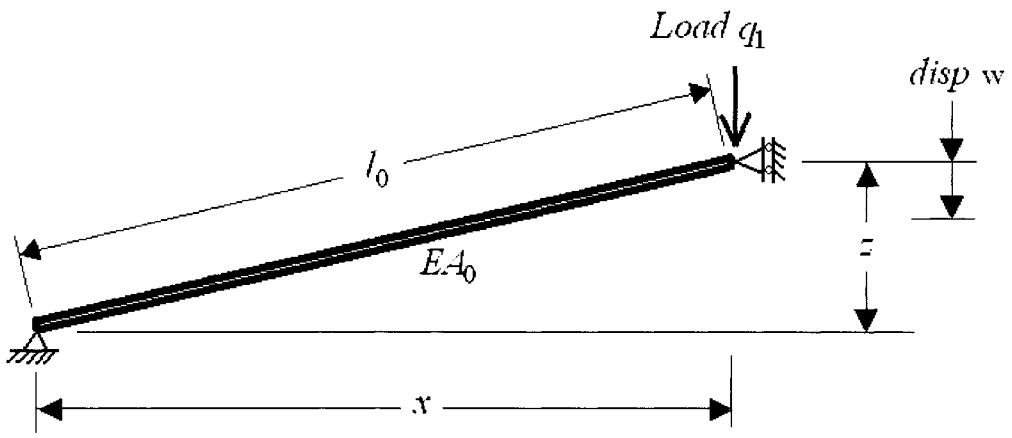


Figure 12. Single-degree-of-freedom truss spring problem.

The exact load displacement path is given in Crisfield [10] as:

$$q_i = \frac{EA_0 Z(2Zw - w^2)}{2l_o^3} \tag{4.1}$$

where $Z = (z + w)$ (4.2)

and E =Young's modulus, A_0 =area and q_i =load increment

The finite element model used for verification is shown in figure 13. Restraints are provided to give free movement in the vertical direction on the right node. The model's elastic modulus is 10e6 $\#/in^2$ and the truss cross sectional area is 5 in^2. A vertical downward load of 100 # is applied at the second node. The truss model was also solved using the exact equation (4.1) by letting $EA_0 = 5.e7\#$, $x = 2500.in$, $z = 25.in$, $q_1 = 100.$

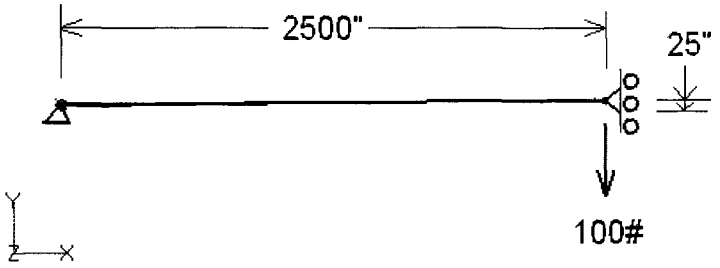


Figure 13. Crisfield single-degree-of-freedom model.

Results from the exact solution, proposed algorithm and NASTRAN are shown in the load-displacement plots in figure 14. The results in [20] showed that the truss tangent stiffness given by equation 3.30 was behaving properly for large geometric nonlinear responses. Figure 14 shows three equilibrium paths that are nearly the same. A numerical comparison of the four points (A thru D) shown in figure 14 are listed below in table 1.

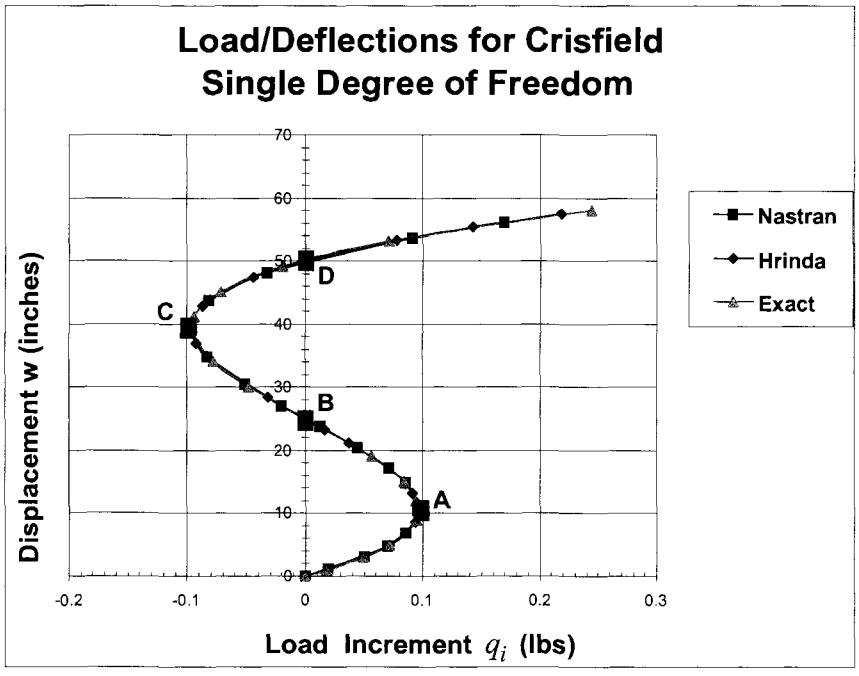


Figure 14. Load increment vs. displacement for the sdof problem.

Appendix A. 4 gives the computer model input file and a more detailed listing of the results plotted in figure 14. The vertical deflections for the sdof problem during snap-through are shown in the sequence of plots in figure 15.

Table 1. Quantitative results comparison of the sdof problem using NASTRAN, the finite element program by Hrinda [20] and the exact solution given by Crisfield [10]. A thru D are the data (load increment, displacement) associated with the curve in figure 14.

METHOD	**A**	**B**	**C**	**D**
NASTRAN	(.0935, 12.59)	(0.000, 24.82)	(-.0962, 39.26)	(.0077, 50.38)
Hrinda	(.0948, 12.00)	(0.000, 24.97)	(-.0961, 39.00)	(.0070, 50.34)
Exact	(.0948, 12.00)	(0.000, 25.00)	(-.0962, 39.26)	(.0077, 50.38)

This example demonstrates a classical snap-through equilibrium path that has been investigated in many [14], [15] and [24]. Further results for this example are presented in appendix A.5. In these results, changes in tangent stiffness and the slope of the equilibrium path (load vs. displacement) are plotted to show their behavior during limit points and snap-through events. The tangent stiffness reduces to zero at the start of instability and then becomes negative during an unstable snap-through.

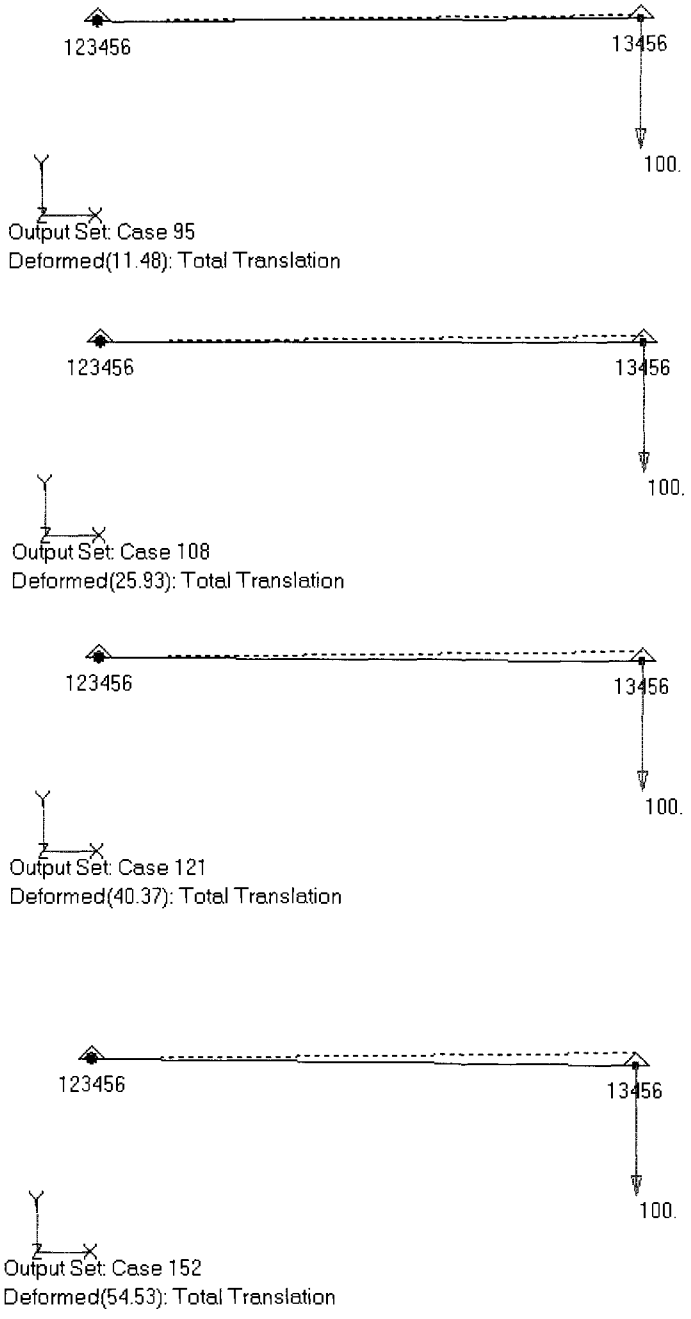


Figure 15. Single-degree-of-freedom deflections. Values given are for the right node with vertical displacements in inches.

4.2 Star Dome Truss

The Crisfield star dome model was taken from Crisfield [11] and has been studied by others to demonstrate a complex equilibrium path. The 3D shallow dome is shown in figure 16 with the two axes of the Cartesian co-ordinate system and the normal z axis indicating the out-of-plane dimensions of the nodes. The model differs from the [9] example by having one concentrated load acting downward at the center node. This load produces multiple snap- through and snap-backs as shown in figure 20. The shape of this equilibrium path is comparable with the equilibrium path found by Geers in [25]. This model is shown in Appendix A.6.1 with its "looping" equilibrium path that has the same shape as the Crisfield star dome equilibrium path in figure 20. Appendix A.6 gives the model input file and a listing of the output data used in figure 20.

The Crisfield star dome was solved using the Hrinda arc length computer program with results compared to a NASTRAN model. The load increment vs. vertical displacements of the center node are plotted in figure 20 and compared. This model introduced a major difficulty following the load path at snap points. The proposed algorithm was able to accurately follow the NASTRAN results and through several snap-through points. Figures 17-19 show the NASTRAN model's response along the equilibrium path of figure 20. Points "A-J" are important to mention in figure 20 because they define some key principles in understanding stability equilibrium paths. Points A, B, C, E, F, H, I and J all have zero tangent stiffness and represent an unstable structure.

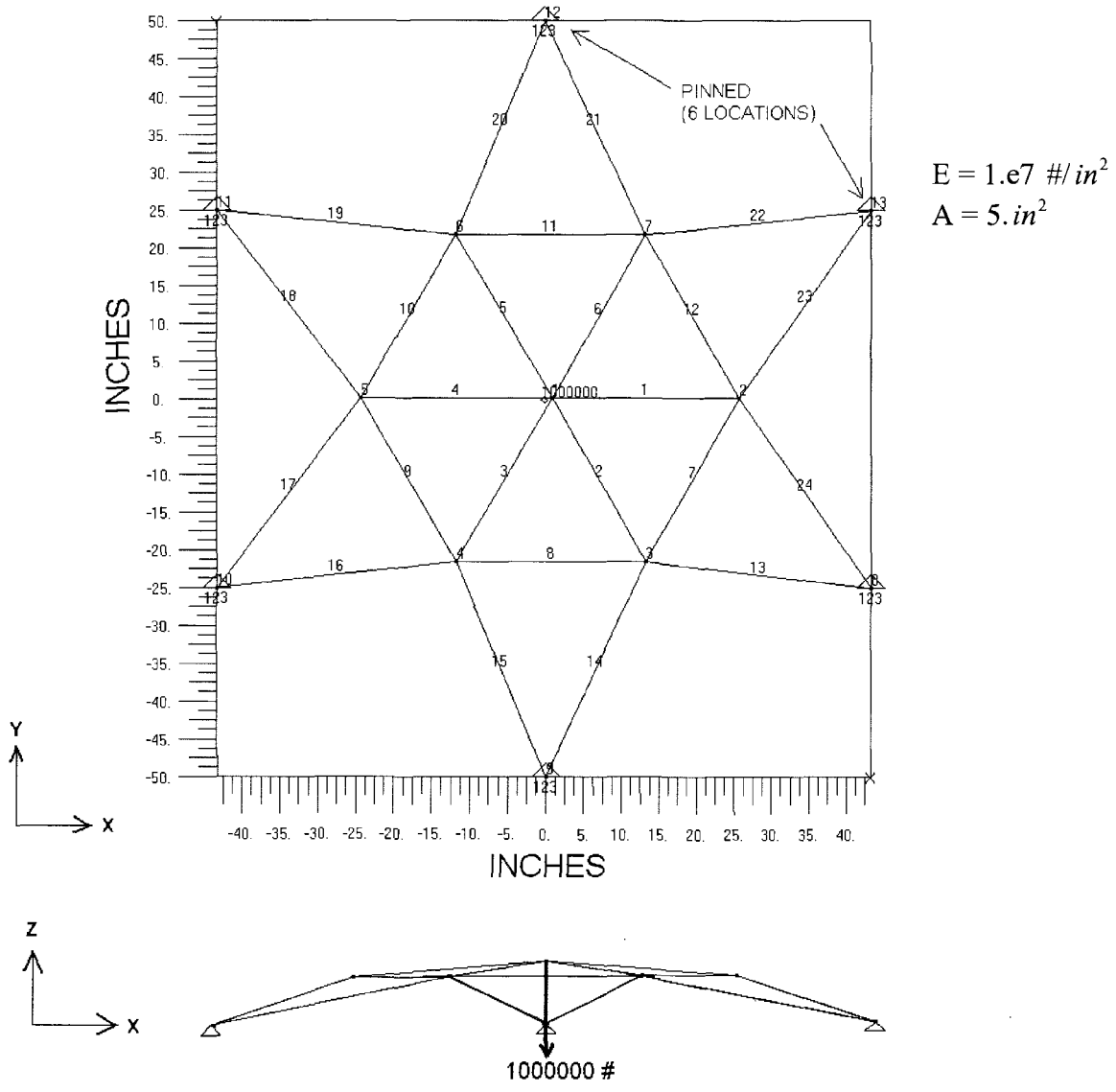
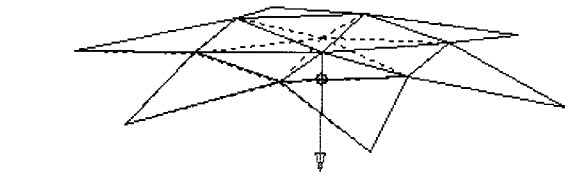


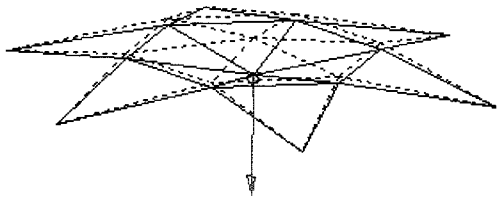
Figure 16. Crisfield three-dimensional star dome using 24 three dimensional truss elements, 13 nodes (39 dofs) and one concentrated load acting vertical downward at the center with the six outer nodes pinned.

Using figure 20, the first snap-through starts at A and continues past B until the same load at A is reached but at a much larger displacement. Figure 20a is a magnified view of the snap-through shown in figure 20. The magnified view shows the first limit point, A, and then an unstable snap-through until A'. The dashed line from A to A' represents the unstable part of the equilibrium path. Point A' is stable and marks the end of the first dynamic snap-through of the structure. Figure 17 shows the displacements of the inner six elements as they snap-through. Once at A', the structure is once again stable and stationary. This is a second stable equilibrium state with the structure in a much different configuration. The applied load may now either incrementally increase from A' to the second limit point, C, as seen in figure 20, or just remain at the current load increment. If the applied load is allowed to continue then the tangent stiffness will become zero. Any infinitesimal increase in the applied load beyond C will cause the structure to begin another unstable snap-through. The unstable equilibrium path in figure 20 continues with the load decreasing and eventually reversing direction. The tangent stiffness becomes negative along this path as the structure deforms. The sequential plots of the structural displacements in figure 18 show that the outer elements, 13-24, are the main contributors to the second snap-through. Instead of immediately snapping through to point C' in figure 20b, the structure begins a snap-back. Here it should be mentioned that the applied dead load on the structure is the value at C. The next stable point on the path, C', will also be at the same applied dead load as C. At D in figure 20, the equilibrium path begins to turn caused by the deflections reversing direction. In this case, D is where the tangent to the equilibrium path is vertical and a snap-back begins. The inner group of elements (1-6) produces the snap-back that can be seen in the last two displacement plots in figure

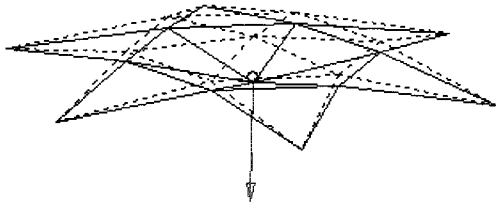
18. These plots show that the structure is rebounding upward starting with the inner ring of elements. As mentioned before, the equilibrium path will become stable at C' but must first pass through an unstable path characterized in figure 20 points D through J. At E, the tangent slope is parallel to the displacement axis as the snap-back continues through point F to point G. In figure 20, G is the limit of the snap-back and is shown in the last displacement plot in figure 18. The structure begins to displace downward again with the center ring of six elements undergoing the most movement. The displacement plots in figure 19 show these members deforming as the unstable equilibrium path in figure 20 is followed to point C' in figure 20b. A closer inspection of the unstable path reveals that the load pattern similar to the first snap-through event is present near point I in figure 20b. This represents the inner ring of elements reaching their maximum snap-through. They start pulling the outer ring of elements, 13-24, downward as shown in the last displacement plot in figure 19. Figure 20b shows that once the load becomes positive at I', the structure has another snap-through from point I to I''. The stable equilibrium point C' is next reached at the same applied dead load as C but with the structure in a completely different configuration. This example shows that a structure may become stable after reaching its first limit point. Another interpretation is that some structures may become stable after reaching a new buckled configuration. The purpose of this example was to demonstrate the effectiveness of the proposed Hrinda program by finding limit points and accurately tracing highly nonlinear equilibrium paths. The interest in this work is mostly focused on the first limit point and how to optimize a structure prone to snap-through. This will be investigated with a larger star dome example in chapter 8.



Output Set: Case 22
Deformed(2.678): T3 Translation



Output Set: Case 228
Deformed(7.395): T3 Translation



Output Set: Case 368
Deformed(9.237): T3 Translation

Figure 17. Star dome deflections. Values given are for the center node with vertical displacements in inches.

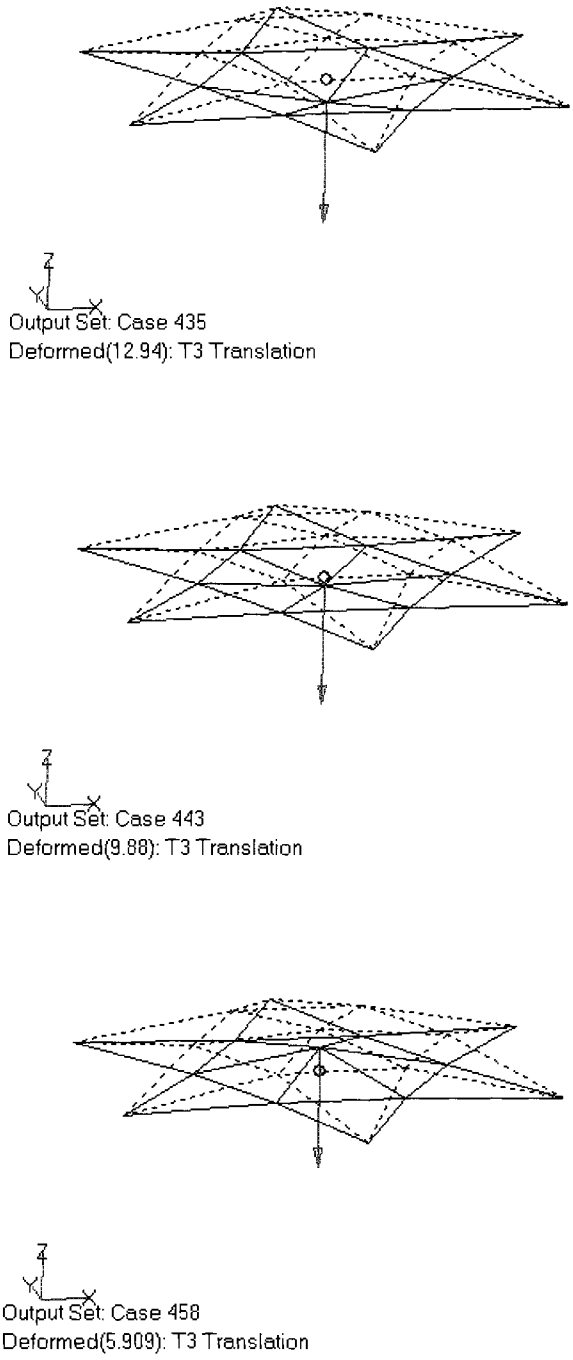


Figure 18. Star dome deflections. Values given are for the center node with vertical displacements in inches.

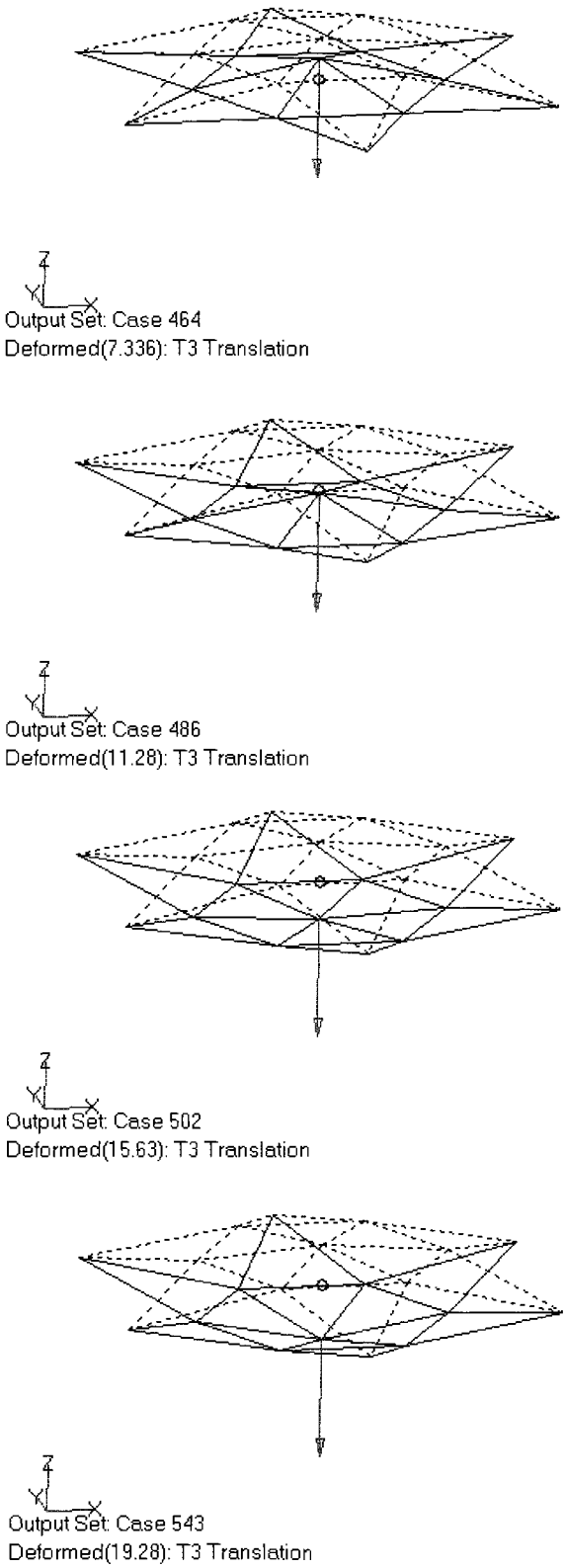


Figure 19. Star dome deflections. Values given are for the center node with vertical displacements in inches.

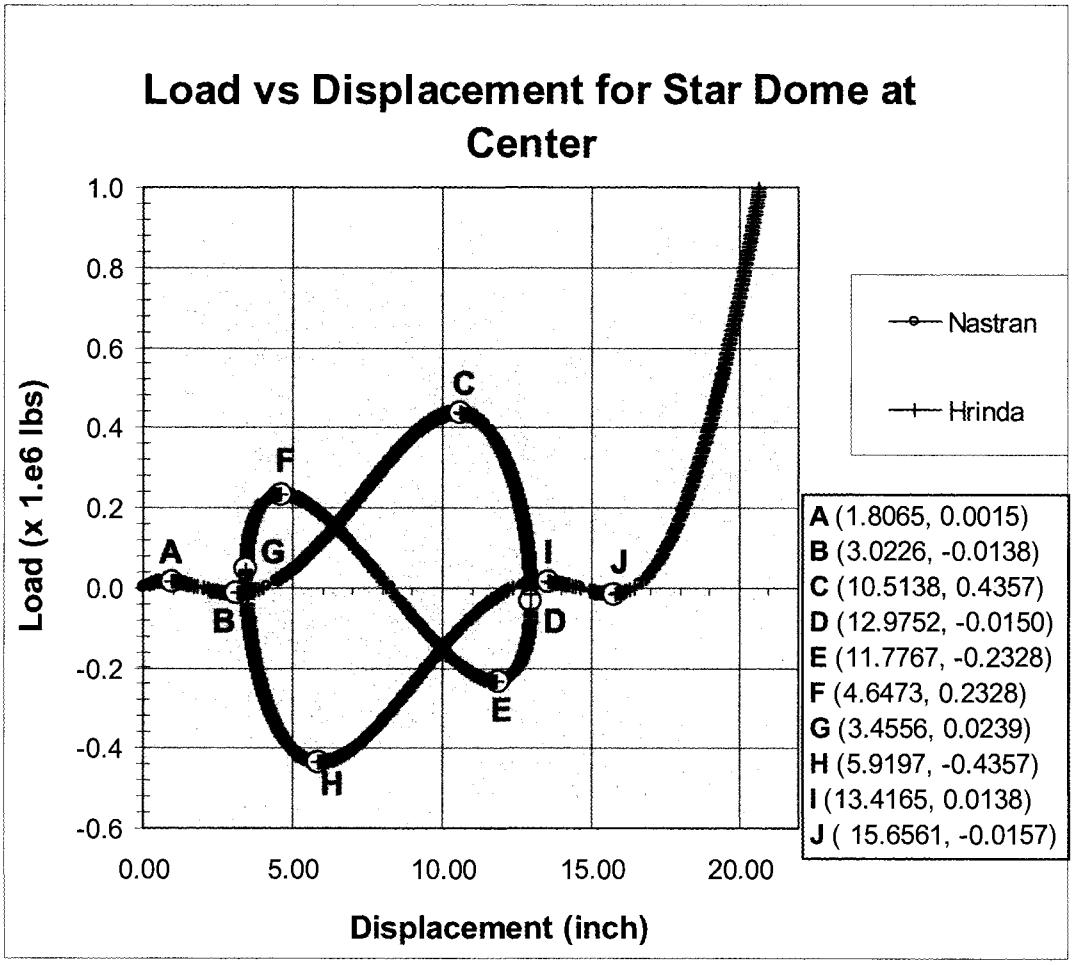


Figure 20. NASTRAN/Hrinda star dome result at center load increment vs. vertical displacement.

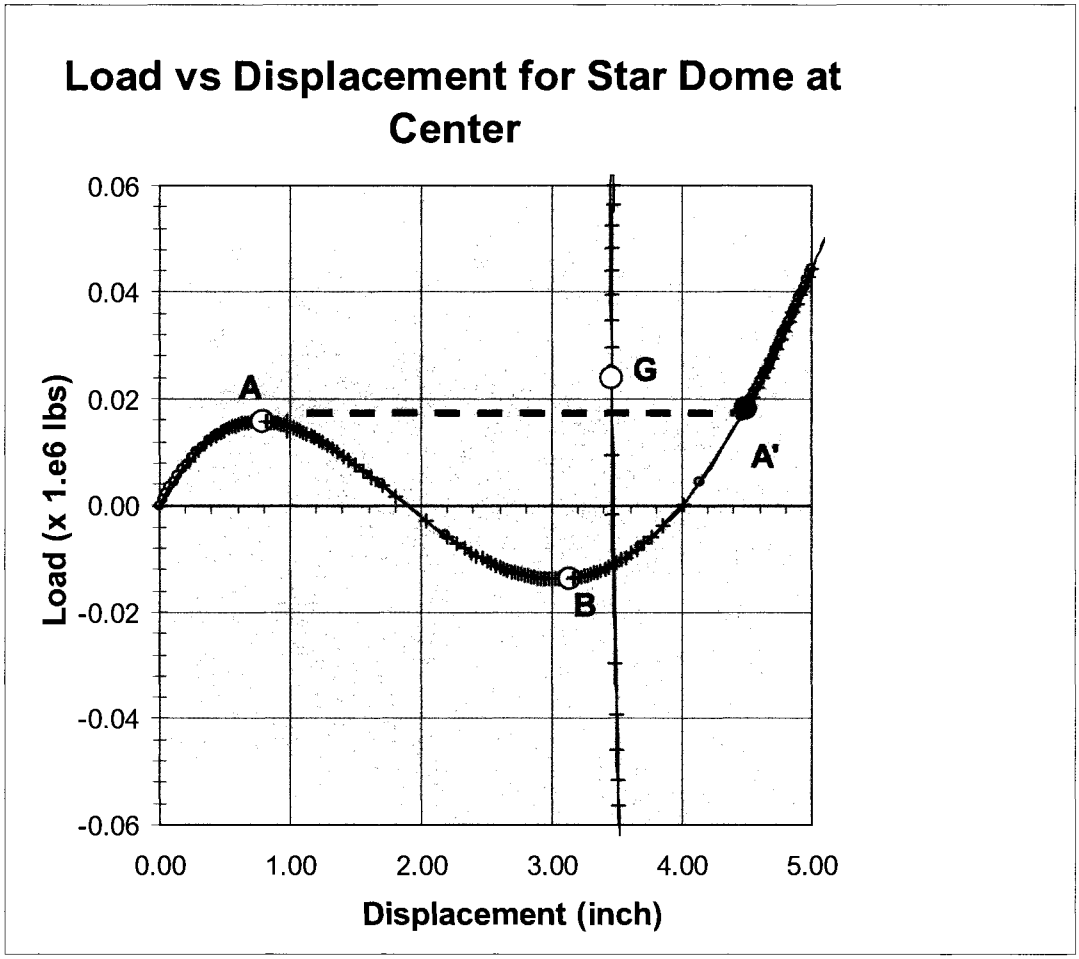


Figure 20a. NASTRAN/Hrinda star dome magnified results of first snap-through from A to $A^{'}$.

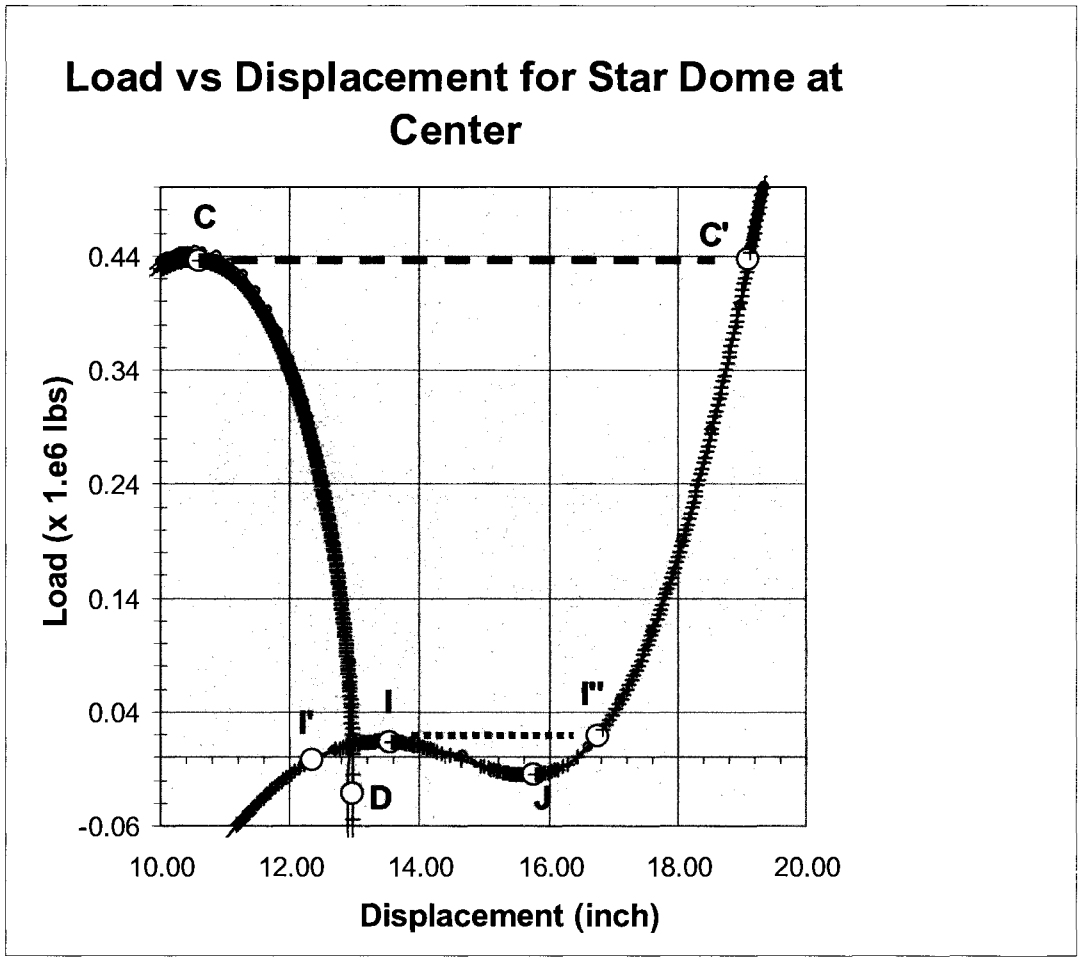


Figure 20b. NASTRAN/Hrinda star dome magnified results of second and third snap-through.

4.3 Crisfield Arch Truss

The model is taken from Crisfield [11] and tests multiple snap-through and snap-back equilibrium paths. The model has 101 elements with 42 nodes with a total of 126 degrees-of-freedom. The out-of-plane motion has been constrained with pin supports added to each end of the truss. Figure 21 shows the test model and the applied load of 1e7 lbs at the apex. Figures 22 through 24 show the deflections from NASTRAN during multiple snappings of the arch. Figure 25 displays an abbreviated plot of NASTRAN results and the predicted response of the Hrinda arc length computer program. Appendix A.7 has the input file for this model and a listing of results used in figure 25. Appendix A.7.1 shows a typical equilibrium path for arches from [26]. The paths shown in the reference are very similar to the looping patterns in figure 25.

Referring to figure 25, the first snap-through, point A, starts when the tangent to the equilibrium path is parallel to the displacement axis. The arch becomes unstable and releases stored strain energy and dynamically traces the equilibrium path from A to B in figure 25. The deformed structure with the vertical apex node values is shown sequentially in figure 22. A snap-back starts at B and continues through C and on to D. This is seen in the deformed plots in figure 23. The snap-back is still unstable and occurs when the tangent to the equilibrium path is parallel to the load axis. The snap-back continues dynamically until D in figure 25 is reached. The equilibrium path now continues to E where another unstable snap-through begins. This is seen in the deformed plots in figure 24. The arc length program closely follows the equilibrium path found by NASTRAN.

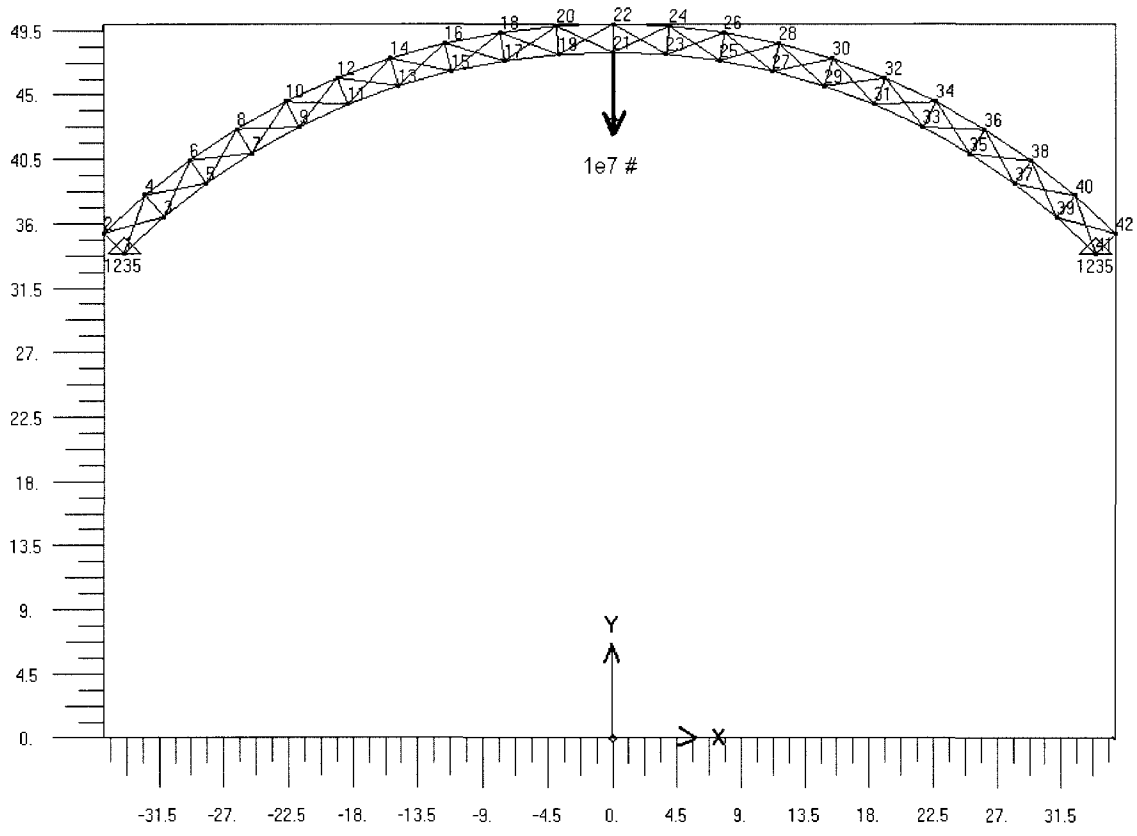
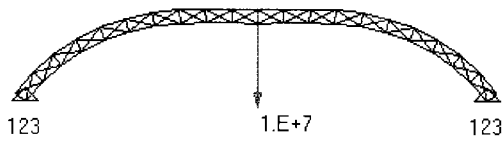
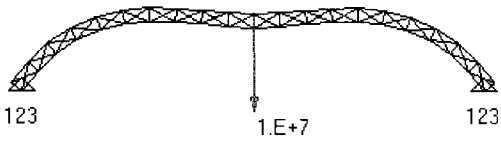


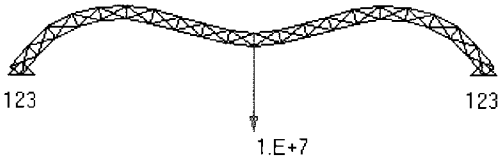
Figure 21. Crisfield large circular arch.



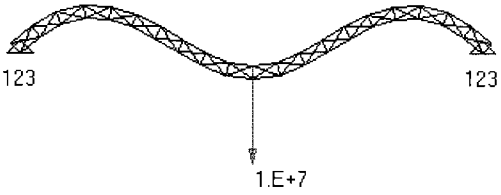
Output Set: Case 30
Deformed(2.619): Total Translation



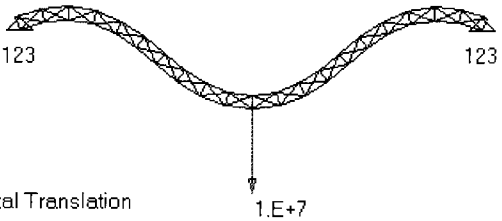
Output Set: Case 50
Deformed(4.752): Total Translation



Output Set: Case 75
Deformed(9.827): Total Translation



Output Set: Case 100
Deformed(17.94): Total Translation



Output Set: Case 125
Deformed(25.75): Total Translation

Figure 22. Arch deformations. Values given are for the apex node with vertical displacements in inches.

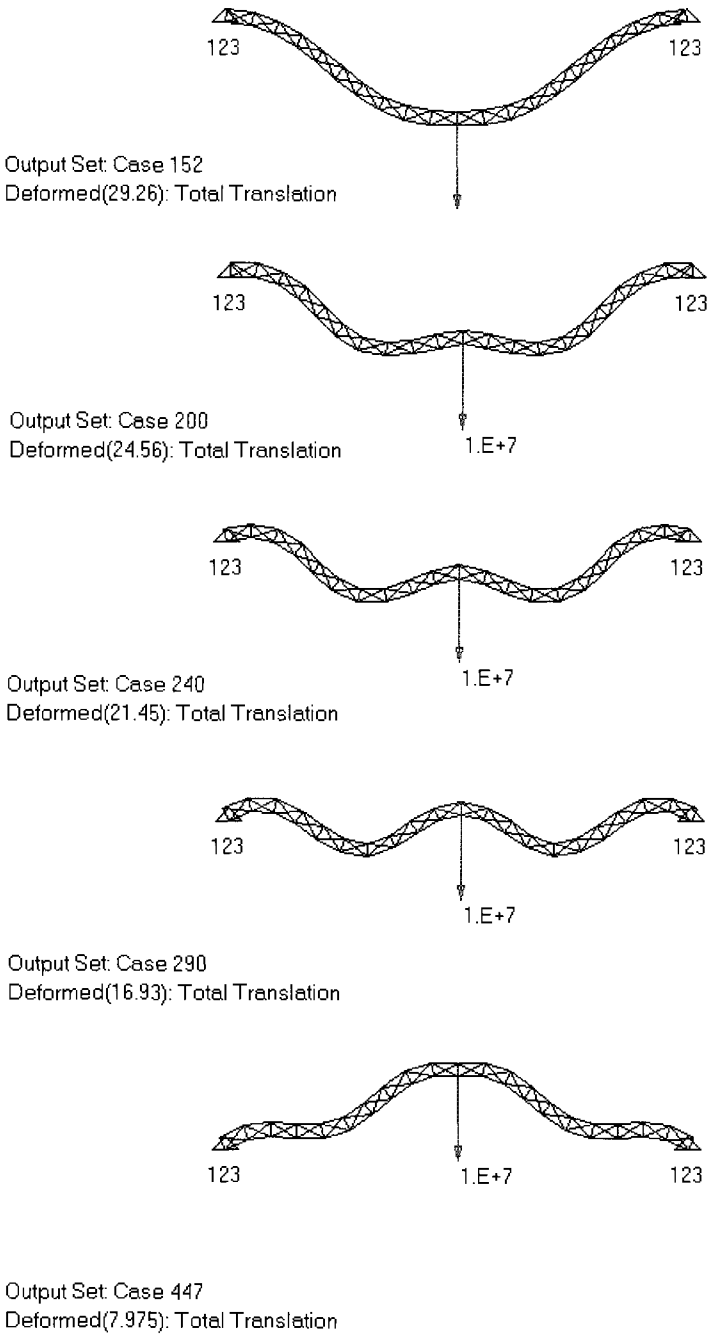


Figure 23. Arch deformations. Values given are for the apex node with vertical displacements in inches.

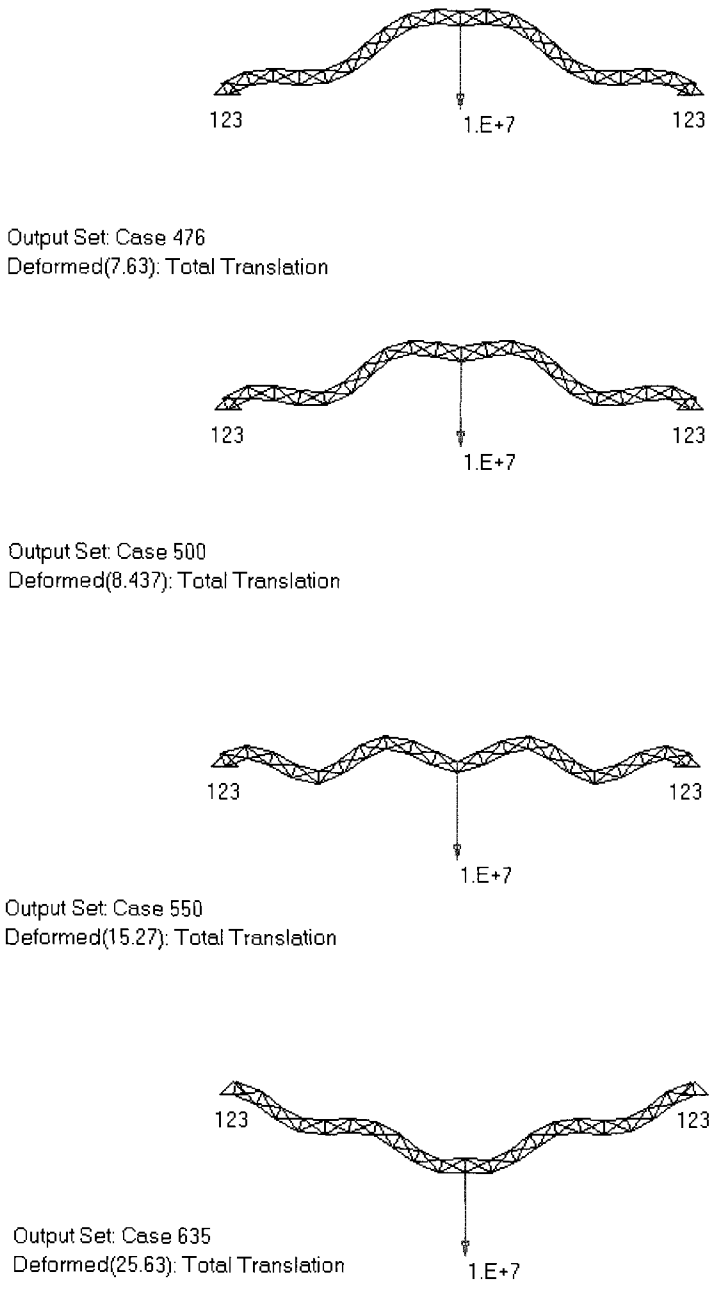


Figure 24. Arch deformations. Values given are for the apex node with vertical displacements in inches.

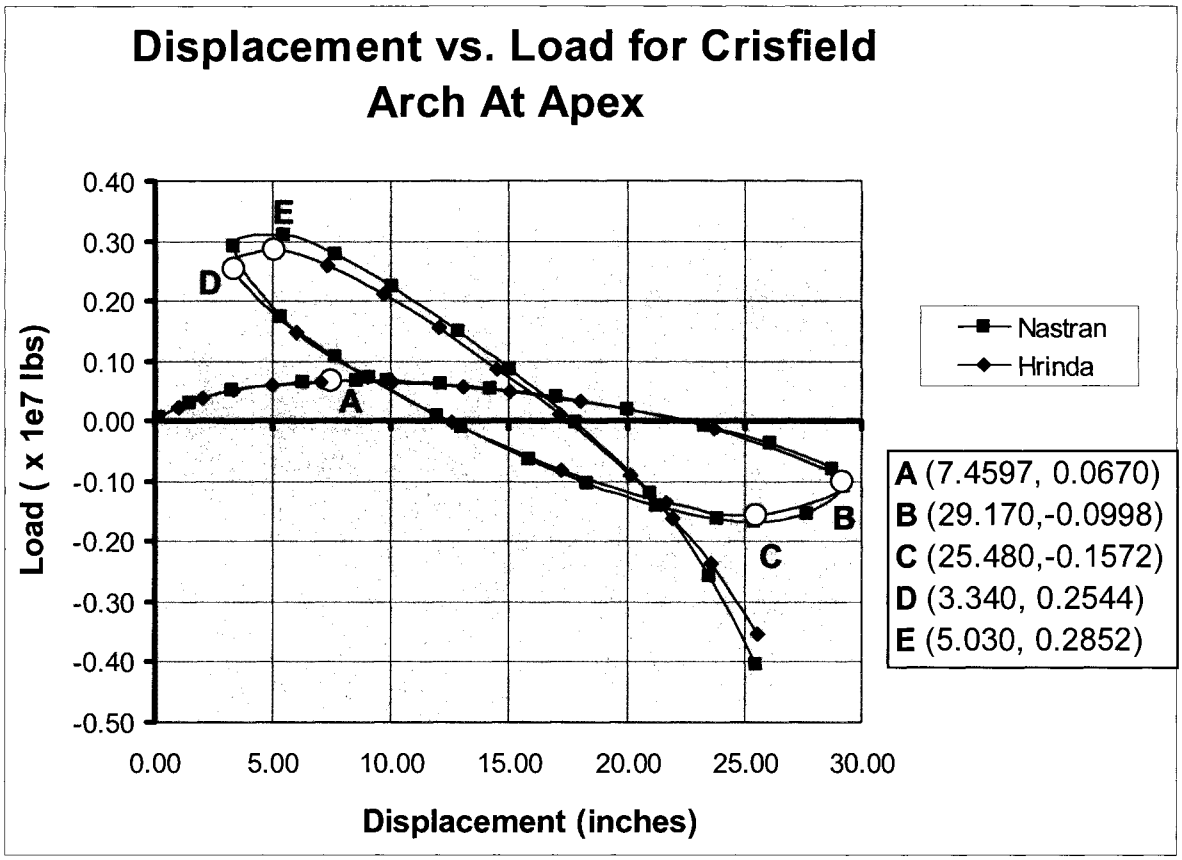


Figure 25. Arch Load Deflection apex comparisons of NASTRAN vs. Hrinda.

CHAPTER V

SPARSE COMPUTING TECHNIQUES

The stiffness matrix is used to describe the members' stiffness relations for computing member forces and displacements in a structure. The diagonals of the matrix are real terms with only a few non-zero, off-diagonal terms. The matrix is mostly populated with zeros and is referred to as "sparse." The stiffness matrix is inverted and multiplied by a force array to give displacements. As the stiffness matrix increases in size, it becomes more time consuming to solve the system of equations for the unknown displacements. Also, the computer storage requirements for storing and inverting a full symmetric stiffness matrix could be too large with the lengthy computation time that prevents efficient use of computer resources. To overcome these computing problems and efficiently utilize the storage capacity of the computer system, the symmetric stiffness matrix information is stored in several single column arrays. The technique only stores the non-zero stiffness terms and thus reduces the computer storage requirement as well as the solution time. The sparse storage techniques in this work are from Nguyen in [19] and are programmed into the proposed algorithm.

5.1 Sparse Storage

The fundamental equations used in structural analysis are written in matrix form as

$$[K]\{Z\} = \{F\} \tag{5.1}$$

where $[K]$ is the structural stiffness matrix, $\{Z\}$ are the unknown displacements and $\{F\}$ are the applied forces. The system is usually symmetric and positive definite but may involve a large stiffness matrix. The size of the stiffness matrix will affect how fast the

system will be solved as well as the amount of computer storage required. The stiffness matrix is usually populated with many zero terms and only a few terms that describe the stiffness. These matrices are called "sparse" and are a major source of computation resources [19].

The process for reading in a sparse stiffness matrix from a finite element model is illustrated through the four bars sample given by Khot and Kamat in [13] and shown in figure 26. The 3D truss structure has 4 elements and 5 nodes with 15 degrees-of-freedom. There is one applied load acting in the negative z direction at node 5. The structure is pin supported at nodes 1 through 4. The nondimensional nodal coordinates are given in table 2.

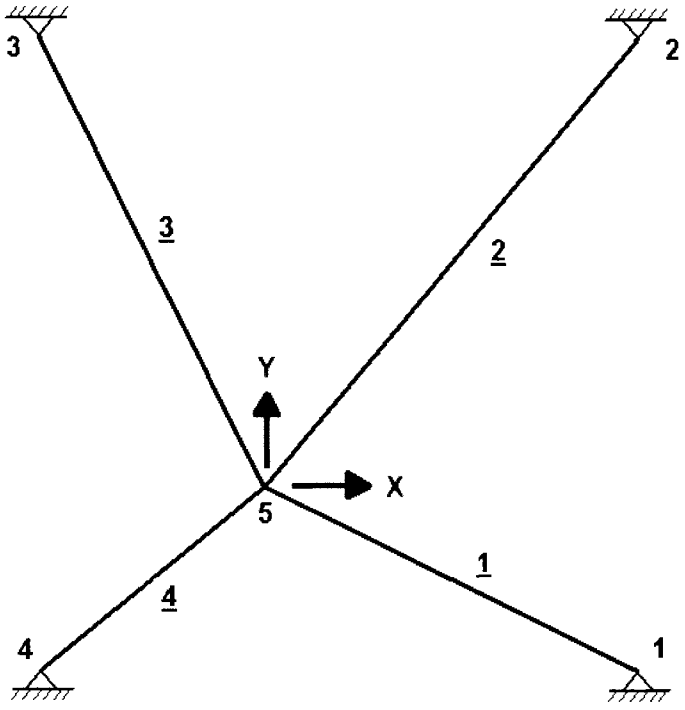


Figure 26. Khot four bar truss.

Table 2. Khot four bar truss nodal coordinates.

NODE	X	Y	Z
1	120.0	-60.0	0.0
2	120.0	144.0	0.0
3	-72.0	144.0	0.0
4	-72.0	-60.0	0.0
5	0.0	0.0	3.0

The global dof associated with each element can be shown in the following arrays

$$lm = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 13 \\ 14 \\ 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \\ 13 \\ 14 \\ 15 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 9 \\ 13 \\ 14 \\ 15 \end{bmatrix}, \begin{bmatrix} 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{bmatrix}$$

$$locbegin = 0, \quad 6, \quad 12, \quad 18$$

Figure 27. Element connectivity arrays.

The vertical load is represented in array "b" as shown in (5.2).

$$b = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ -1.0 \end{bmatrix} \tag{5.2}$$

The four pinned constraints are given in array "iboundc" (5.3).

$$iboundc = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.3}$$

The global stiffness can now be written as a 15 x 15 symmetric matrix and is shown in figure 28. The matrix is obtained from the four element arrays in figure 27 and the element stiffness in chapter 3. The arrays used to efficiently store the stiffness matrix in sparse form will now be discussed.

1	0	0	.	.	0	0	0	1
0	1	0	.	.	0	0	0	2
0	0	.....	.	.	.	.	.	.
.	.	.	.....	.	.	.	.	.
.	.	.	.	.....	.	.	.	.
.	.	.	.	.	156728.0	52393.59	1101.028	13
.	.	.	.	.	52393.59	113969.42	1237.511	14
0	0	0	.	.	1101.028	1237.511	181.6088	15
1	2	3	.	.	13	14	15	

Figure 28. Khot four element truss stiffness matrix after applying restraints.

The element-dof connectivity information is given in the following two arrays, "ia" and "ja." Both arrays are constructed using the element conductivity arrays from figure 27. Array "ia" gives the locations of the first non-zero of each row of the element dof connectivity and is shown in (5.4).

$$ia = \begin{bmatrix} 1 \\ 7 \\ 13 \\ 19 \\ 25 \end{bmatrix} \tag{5.4}$$

Array "ja," given in (5.5), contains the global dof column numbers for the non-zero terms of each row of the element dof connectivity.

$$ja = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 13 \\ 14 \\ 15 \\ 4 \\ 5 \\ 6 \\ 13 \\ 14 \\ 15 \\ 7 \\ 8 \\ 9 \\ 13 \\ 14 \\ 15 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{bmatrix} \tag{5.5}$$

The values of the diagonal stiffness terms are then stored in "ad" given in (5.6).

$$ad = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 156728.0469 \\ 139694.1853 \\ 181.6088 \end{bmatrix} \tag{5.6}$$

The stiffness values of the non-zero, off-diagonal terms are stored in array "an" and given in (5.7).

$$an = \begin{bmatrix} 52393.5876 \\ 1101.0280 \\ 1237.5111 \end{bmatrix} \tag{5.7}$$

The technique for storing the sparse matrix requires 15 diagonal storage terms and only 3 storage terms for the off-diagonal terms.

5.2 Sparse Solver

In order to solve for the unknown displacements in equation (5.1), the Choleski numerical factorization method [19] is used with the sparse matrix previously described. The method has been extensively utilized within commercial finite element analysis

programs using sequential computers. The Choleski method consists of three computation steps that efficiently solve a system of linear equations by preventing the storage of a full square matrix. The steps are programmed into the proposed algorithm using sparse techniques that further increase solution efficiency. The first step starts with decomposition of

$$[K] = [U][U]^T \tag{5.8}$$

where $[U]$ is a lower factorized triangular matrix and the transpose, $[U]^T$, is the lower factorized triangular matrix. The diagonal terms of $[U]$ are defined as

$$u_{i,i} = \sqrt{k_{i,i} - \sum_{k=1}^{i-1} u_{k,j}^2} \tag{5.9}$$

and the off-diagonal terms as

$$u_{i,j} = \frac{a_{i,j} - \sum_{k=1}^{i-1} u_{k,j} u_{k,j}}{u_{i,j}} \tag{5.10}$$

where j>i.

The next step is a forward solution that begins by substituting $[U]$ into equation (5.8) to give

$$[U]^T [U][Z] = [F] \tag{5.11}$$

This equation can be altered by substituting an unknown vector $[y]$ for the product of $[U][Z]$ in equation (5.11) that leads to

$$[U]^T [y] = [F] \tag{5.12}$$

The forward solution of equation (5.12) yields the vector $[y]$ that is used in the next step.

The third and final step combines the factorized matrix $[U]$ and the vector $[y]$ from step two to obtain

$$[U][Z]=[y] \tag{5.13}$$

This leads to the final solution for the unknown displacements $[Z]$ in equation (5.1).

5.3 Symbolic Sparse Factorization

The efficiency of a sparse equation solution can be improved through symbolic sparse factorization. During the factorization phase for solving equation (5.11) there will be extra non-zero "fill-in" terms. The symbolic factorization will find the locations of all the non-zero off-diagonal terms in the factorized matrix $[U]$. For the four member truss example used in this chapter, two arrays are used to store the location information. The first array, shown in (5.14), is "ia" and gives the stating locations of the first non-zero, off-diagonal terms in the ith row of the factorized matrix $[U]$. The second array, given in (5.15), is "ja" and contains the column numbers corresponding to each nonzero, off-diagonal term in the ith row of $[U]$.

$$ia = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 4 \\ 4 \end{bmatrix} \tag{5.14}$$

$$ja = \begin{bmatrix} 14 \\ 15 \\ 15 \end{bmatrix} \tag{5.15}$$

Numerical factorization is performed inside the subroutine "numfa1d" found in appendix C. The inverse of the diagonal factorized matrix $[U]$ is given as

$$di = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 6.3804E-6 \\ 8.1846E-6 \\ 5.9634E-3 \end{bmatrix} \tag{5.16}$$

The numerical values of the non-zero, off-diagonal terms in the factorized matrix $[U]$ are given as

$$un = \begin{bmatrix} 0.3342 \\ 7.0250E-3 \\ 7.116E-3 \end{bmatrix} \tag{5.17}$$

The last step of the sparse equation solver is the forward/backward solution performed by equations (5.11) through (5.13) and programmed into subroutine "fbed" in the Hrinda algorithm. The upper triangular matrix of the factorized matrix $[U]$ is input to the subroutine using the arrays defined. The solution is the unknown displacement array at the given input load increment.

CHAPTER VI

DERIVATION OF OPTIMUM NONLINEAR STRAIN ENERGY DENSITY FORMULAS

The discussion of using the strain energy density as an indicator of an optimal design begins with an explanation of the strain and potential energy in truss elements. The optimization problem can then be defined with a recurrence formula that is developed to update the design variable--in this case, the cross-sectional area of the member.

Green's strain is used to develop the strain energy density relationship in the 3-D truss element. This is the same strain measurement used in developing the tangent stiffness derived in chapter 3 and shown in equation (3.30). A truss element's mass density is included in the strain equation to obtain the level of strain energy in the element. This section begins by explaining the formulation of Green's strain and then the derivation of the strain energy with strain energy density to show the relationship in equation (6.18).

This relationship is used in developing the recurrence formula that modifies the cross-sectional area to reach a minimal weight design. Obtaining an updating recurrence relationship is possible by formulating a nonlinear constrained optimization problem that uses the potential energy as an equality constraint. The first step to solving the nonlinear constrained optimal problem and deriving an update scheme for the area is to form the Lagrangian. The method will demonstratc how to minimize the strain energy density while making sure the structure supports a load without becoming unstable. The method is the basic tool used in nonlinear constrained optimization problems. Section 6.2 illustrates the derivation of the optimality criterion using the potential energy as an

equality constraint and how it is incorporated into the Lagrangian. The discussion proceeds with differentiating with respect to the design variable to obtain the minimum potential energy. The strain energy density (SED) is introduced to obtain a Lagrange multiplier γ given in equation (6.49) and reproduced below:

$$\gamma_i = \frac{\sum_{i=1}^{n} SED}{\sum_{i=1}^{n} SED^2} \tag{6.1}$$

The Lagrange multiplier is placed into the recurrence formula (6.45) to obtain the final form of the recurrence formula given by equation (6.50) and also shown below:

$$(A_i)_{\text{iter+1}} = (A_i)_{\text{iter}} \left(\frac{\sum_{i=1}^{n} SED}{\sum_{i=1}^{n} SED^2} \right) (SED_i)^{\lambda} \tag{6.2}$$

This equation is the recurrence relation that is used to update the design variable A_i at the end of each converged arc length iteration. This difference equation defines each term of the sequence as a function of the preceding terms. The equation gives a new A_i, based on the strain energy density differences in all of the members. As an optimal solution is approached, the strain energy densities will become equal within an allowable user-specified difference.

6.1 Green's Strain

The strain in a truss element may be expressed as

$$\varepsilon_E = \frac{xl_i^* - xl_i}{xl_i} \tag{6.3}$$

where xl is the original length along the local x-axis and xl_i is the new length of the ith member as shown in figure 29 [23].

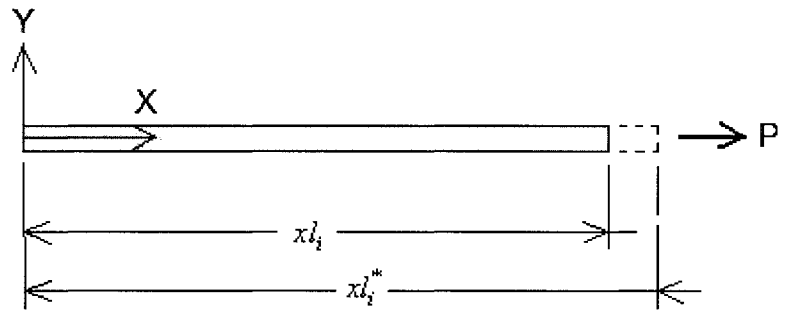


Figure 29. Truss elongation from load P.

Multiplying equation (6.3) by $\frac{xl_i^* + xl_i}{xl_i^* + xl_i}$ will result in

$$\frac{xl_i^* - xl_i}{xl_i}\left(\frac{xl_i^* + xl_i}{xl_i^* + xl_i}\right) = \frac{xl_i^{*2} - xl_i^2}{\left(xl_i\, xl_i^* + xl_i^2\right)} \tag{6.4}$$

Equation (6.4) may be expanded into the following:

$$\frac{xl_i^{*2} - xl_i^2}{\left(xl_i xl_i^* + xl_i^2\right)} = \frac{xl_i^{*2} - xl_i^2}{\left(xl_i xl_i^* + 2xl_i^2 - xl_i^2\right)} \tag{6.5}$$

Further manipulation of equation (6.5) is performed as shown in the following sequence:

$$\frac{xl_i^{*2} - xl_i^2}{\left(xl_i xl_i^* + 2xl_i^2 - xl_i^2\right)} = \frac{xl_i^{*2} - xl_i^2}{\left(2xl_i^2 + xl_i(xl_i^* - xl_i)\right)} \tag{6.6}$$

$$\frac{xl_i^{*2} - xl_i^2}{\left(2xl_i^2 + xl_i(xl_i^* - xl_i)\right)} = \frac{xl_i^{*2} - xl_i^2}{\left(2xl_i^2 + xl_i^2(\frac{xl_i^* - xl_i}{xl_i})\right)} \tag{6.7}$$

$$\frac{xl_i^{*2} - xl_i^2}{\left(2xl_i^2 + xl_i^2(\frac{xl_i^* - xl_i}{xl_i})\right)} = \frac{xl_i^{*2} - xl_i^2}{\left(2xl_i^2 + xl_i^2\varepsilon_E\right)} \tag{6.8}$$

$$\frac{xl_i^{*2} - xl_i^2}{\left(2xl_i^2 + xl_i^2\varepsilon_E\right)} = \frac{xl_i^{*2} - xl_i^2}{xl_i^2\left(2+\varepsilon_E\right)} \tag{6.9}$$

For small elastic strains, equation (6.9) may be written as

$$\varepsilon_i = \frac{xl_i^{*2} - xl_i^2}{2xl_i^2} \tag{6.10}$$

The formulation is referred to as "Green's strain" [10].

6.2 Strain Energy Density Formulation

The length change of the element may be large; however, the strains are considered small enough to allow a linear stress strain relationship. Strain energy is stored within an elastic solid when the solid is deformed under load. In the absence of energy losses, such as from friction, damping or yielding, the strain energy is equal to the work done on the solid by external loads. Strain energy is a form of potential energy. The work, W, preformed by a gradually applied load is [23]

$$W = \frac{1}{2}P\Delta \tag{6.11}$$

where P is the internal load and Δ is the axial displacement. It should be noted here that equation 6.11 is still valid for nonlinear, large and finite displacement considered in this work since the strain resulting from such a displacement is assumed to be small enough for a linear stress strain relation [13]. The work that is done by external forces to cause deformation is stored within the truss in the form of strain energy. Based on the assumed linear stress strain relation, the strain energy SE_i of a typical truss element can be computed as

$$SE_i = \frac{1}{2} E_i A_i xl_i \varepsilon_i^2 = \frac{1}{2} (\sigma_i = E_i \varepsilon_i) A_i (\Delta_i = xl_i \varepsilon_i)$$

$$SE_i = \frac{1}{2} P_i \Delta_i \tag{6.12}$$

This relationship is valid for systems that have no energy losses from sources such as friction, yielding or damping.

Substituting $P = AE(xl_i^* - xl_i)/xl_i$ for a truss element and $\Delta_i = (xl_i^* - xl_i)$ into the strain energy equation (6.12) yields

$$SE_i = \frac{1}{2} \frac{A_i E_i}{xl_i} (xl_i^* - xl_i)^2 \tag{6.13}$$

Here, A_i is the truss cross-sectional area and E is Young's modulus of the truss material.

The relationship can also be written in the form

$$SE_i = \frac{A_i E_i (xl_i)}{2} \left(\frac{xl_i^* - xl_i}{xl_i} \right)^2 \tag{6.14}$$

An alternative form of writing equation (6.3) is

$$\left(\frac{xl_i^* - xl_i}{xl_i} \right)^2 = \left(\varepsilon_i - \frac{xl_i^* - xl_i}{xl_i} \right) 2 \tag{6.15}$$

This expression is substituted into equation (6.14) to give the strain energy for an *i*th element, which is

$$SE_i = A_i E_i (xl_i) \left(\varepsilon_i - \frac{xl_i^* - xl_i}{xl_i} \right) \tag{6.16}$$

Strain energy density (*SED*) is the amount of strain energy per unit volume. The strain energy density can be found by dividing the strain energy by the volume of an element and can be written as

$$SED_i = \frac{SE_i}{(xl_i)A_i} \tag{6.17}$$

This relationship is modified for use in an optimum criterion by dividing the strain energy by the mass density ρ_i for an ith element to yield

$$SED_i = \frac{SE_i}{\rho_i xl_i A_i} \tag{6.18}$$

To further develop an optimality criterion for stability, a discussion of potential energy follows.

6.3 Optimality Criterion

The minimum weight optimization of a truss system is defined as:

$$W = \sum_{i=1}^{n} \rho_i A_i xl_i \tag{6.19}$$

where ρ_i is the material density of the ith element, A_i is the cross-sectional area of the ith element and xl_i is the new deformed length of the ith element with n number of elements.

The formal optimization problem can be defined as

find $\{A_i\}$ (6.20)

to minimize $\sum_{i=1}^{n} \rho_i A_i xl_i$ (6.21)

subject to $PE - \overline{PE} = 0$ 6.22)

This minimum weight equation is subject to the equality constraint

$$g = PE - \overline{PE} = 0 \tag{6.23}$$

Where PE is the total potential energy of a truss system and $\overline{PE}$ is the total potential energy associated with the optimal design at the nonlinear critical limit load [13]. This equality constraint has the smallest cross-sectional area that is capable of sustaining an applied load just before the structure becomes unstable. This places a constraint on the limit load that is the same as constraining the total potential energy of the structure. For multiple members, the potential energy may be written as

$$PE = \sum_{i=1}^{\text{ele}} SE_i - \sum_{j=1}^{\text{nodes}} W_j \tag{6.24}$$

where SE_i is the strain energy in the ith element and W_j is the work that is produced from an external force. The relationship can also be written for a structural system that consists of n number of truss elements as

$$PE = \sum_{i=1}^{n} SE_i - \vec{u}^T \vec{F} \tag{6.25}$$

where SE_i identifies the strain energy in the ith element, $\vec{u}^T$ is a vector transpose of the global displacements, and $\vec{F}$ is the applied design load vector.

Lagrange developed the tool for solving optimality problems by developing the Lagrangian function which is written as

$$L = f(x) + \sum_{j=1}^{m} \gamma_j h_j(x) \tag{6.26}$$

$$\text{for finding} \qquad x \in R^n \tag{6.27}$$

$$\text{to minimize} \qquad f(x) \tag{6.28}$$

$$\text{subject to} \qquad h_j(x) = 0 \qquad j = 1, 2, \ldots, m \tag{6.29}$$

where γ_j is called the Lagrange multiplier for the solution (x) that lies in some domain. Lagrange multipliers find the maximum or minimum of a function of several variables subject to one or more constraints. This method reduces a problem with n variables with k constraints to a solvable problem with $n + k$ variables and no constraints. The method introduces a new, unknown scalar variable, the Lagrange multiplier, for each constraint and forms a linear combination that involves the multipliers as coefficients. The Lagrange multiplier can be thought of as how strong the constraint is working to get a point to a maximum or minimum. The objective is to find the conditions for some implicit function so that the derivative, in terms of the independent variables of a function, equals zero for some set of inputs. The point x is stationary in space if

$$\frac{\delta L}{\delta x_i} = 0 = \frac{\delta f}{\delta x_i} + \sum_{j=1}^{m} \gamma_j \frac{\delta h_j(x)}{\delta x_i} \qquad i = 1,2,...,n \tag{6.30}$$

and

$$h_j(x) = 0 \qquad j = 1,2,...,m \tag{6.31}$$

A stationary point merely means that the solution (x) causes the derivative of a function to equal zero, which is equivalent to the point at which the function stops increasing or decreasing. In the case presented in this work, the solution (x) is the value of the load that causes the tangent at that point to be parallel to the horizontal axis of figure 4. This is also the point of minimum strain energy density for all members along their equilibrium path. If this formulation is used and applied to minimizing weight in trusses, then the Lagrangian becomes (with $m = 1$, in equation 6.31)

$$L = \sum_{i=1}^{n} \rho_i A_i x l_i - \gamma\left(PE - \overline{PE}\right) \tag{6.32}$$

where γ is the Lagrange multiplier and $\left(PE - \overline{PE}\right)$ is the difference between the total potential energy of the system PE and the target potential energy $\overline{PE}$ of the optimal design at the critical limit load. We desire this difference in potential energy to be a minimum. Differentiating with respect to A_i (and sending the resulting equation to zero) gives

$$\rho_i x l_i - \gamma_1 \left(\frac{\partial PE}{\partial A_i} + \sum_{j=1}^{\text{dof}} \frac{\partial PE}{\partial u_j} \frac{\partial u_j}{\partial A_i} \right) = 0 \qquad i = 1, 2, ..., n \tag{6.33}$$

The potential energy is a function of the set of displacements. If these displacements place the structure in stable equilibrium then the total potential energy has a "stationary" value and the following can be written

$$\frac{\partial PE}{\partial u} = \frac{\partial SE}{\partial u} - F = 0 \tag{6.34}$$

This is the partial of the potential energy from equation (6.25). Using the zero statement of the equilibrium equation (6.34) with equation (6.33), the following simplified form is achieved:

$$\rho_i x l_i - \gamma_1 \left(\frac{\partial PE}{\partial A_i} \right) = 0 \tag{6.35}$$

Remarks:

1. After taking the derivative of the Lagrangian (see equation 6.32) with respect to the truss cross sectional area A_i, the term $\overline{PE}$ do no longer appears in equation 6.35.
2. The Lagrangian in equation 6.32, without the term $\overline{PE}$, can also be interpreted as a multiple objective optimization problem. One would like to minimize the total

weight of a 3D truss structure and at the same time maximize the total potential energy PE (or minimize $-PE$), and the Lagrange multiplier γ can be considered as the "weighting factor" between the two different objects.

Consider now the potential energy in equation (6.25) as a function of the area of the section. A change in the potential energy with respect to a change in the area can be written as

$$\frac{\partial PE}{\partial A_i} = \frac{\partial SE_i}{\partial A_i} - \frac{\partial u_i F}{\partial A_i} \tag{6.36}$$

which can be expanded using (6.16) and (6.25)

$$\frac{\partial PE}{\partial A_i} = \frac{\delta}{\delta A_i} A_i E(xl_i)\left(\varepsilon_i - \frac{xl_i^* - xl_i}{xl_i}\right) - \frac{\delta}{\delta A_i}\vec{u}^T\vec{F} \tag{6.37}$$

The partial derivative with respect to the area A_i is

$$\frac{\partial PE_i}{\partial A_i} = E(xl_i)\left(\varepsilon_i - \frac{xl_i^* - xl_i}{xl_i}\right) \tag{6.38}$$

This may now be multiplied by $\frac{A_i}{A_i}$ to obtain the SE_i relation in the numerator

$$\frac{\partial PE_i}{\partial A_i} = E(xl_i)\left(\varepsilon_i - \frac{xl_i^* - xl_i}{xl_i}\right)\left(\frac{A_i}{A_i}\right) \tag{6.39}$$

The numerator is equation (6.16) and is used to obtain the reduced form

$$\frac{\partial PE_i}{\partial A_i} = \frac{SE_i}{A_i} \tag{6.40}$$

This is possible because the nonlinear strain energy is only a function of the design variable. Equation (6.40) may now be placed into equation (6.35) to form

$$\rho_i x l_i - \gamma_1 \left(\frac{SE_i}{A_i} \right) = 0 \tag{6.41}$$

Dividing both sides of the above equation by $\rho_i x l_i$, the resulting equation can be rewritten as

$$\frac{\gamma_1}{\rho_i x l_i} \left(\frac{SE_i}{A_i} \right) = 1 \tag{6.42}$$

Using the definition of strain energy density, this equation can be given as the strain energy density optimality criterion of

$$\gamma_1 \left(SED_i \right) = 1 \tag{6.43}$$

Remarks:

1. Even for the nonlinear large displacement case, strain energy is still a linear function of design variables at cross section are A_i. However, the above statement is not true if design variables as "dimensions of cross sectional areas", such as $A_i = b_i * h_i$ where b_i and h_i represent the base and height of a typical cross sectional member, respectively.
2. Since cross sectional areas A_i are used as design variables, scaling the areas and the load parameter by the same scale factor will not change the displacement pattern.

A recurrence formula for updating the area design variable is found by multiplying this formula by $(A_i)^{\beta}$ to give

$$(A_i)^{\beta} = (A_i)^{\beta} (\gamma_1) SED_i \tag{6.44}$$

where β is a parameter that controls the step size and the convergence rate.

Now, taking the β th root on each side

$$(A_i)_{\text{iter+1}} = (A_i)_{\text{iter}} (\gamma_1 SED_i)^{1/\beta} \tag{6.45}$$

gives the updated area for the next iteration. The use of this equation within the arc length method will set $1/\beta$ equal to the load increment multiplier λ_i, as shown in figure 8. The Lagrange multiplier γ_1 in equation (6.45) is unknown but may be found by utilizing equation (6.43) and recognizing that γ_1 will equal 1 at an ideal optimal [18]. Using equation (6.43), a residual may be defined as

$$Res_i = 1 - \gamma_1 (SED_i) \tag{6.46}$$

The residual for an ith member is the amount of estimated error in the member's sizing from the optimal. In statistics, the mean squared error is often used to analyze the performance of an estimated value from the true value. This may be applied to equation (6.46) by taking the sum of the residuals squared written as

$$\frac{1}{n}\sum_{i=1}^{n}(1-\gamma_1 SED_i)^2 \tag{6.47}$$

where

i is the member number and n is the total number of members in the structure. The value of equation (6.47) is desired to be minimal. This can be accomplished by taking the partial derivative of the sum of the squares with respect to γ_1 which is written as

$$\frac{\delta}{\delta\gamma_1}\left[\frac{1}{n}\sum_{i=1}^{n}(1-\gamma_1 SED_i)^2\right] = 0 \tag{6.48}$$

The solution for γ_1 is the elementary function

$$\gamma_1 = \frac{\sum_{i=1}^{n} SED_i}{\sum_{i=1}^{n} SED_i^2} \tag{6.49}$$

and is placed into equation (6.45) along with λ to obtain the recurrence relation

$$(A_i)_{\text{iter+1}} = (A_i)_{\text{iter}} \left(\frac{\sum_{i=1}^{n} SED_i}{\sum_{i=1}^{n} SED_i^2} \right) (SED_i)^{\lambda} \tag{6.50}$$

CHAPTER VII

DESIGN-VARIABLE SCALING

The arc length method is used in the Hrinda optimization algorithm to both solve the nonlinear structural system and also as an update indicator to monitor the strain energy density in the truss elements of the system. As the equilibrium path is defined, parameters of the arc length are monitored and used to update the recurrence equation. The approach allows for updating the truss stiffness after each run with the arc length solver. As a critical-limit point is approached, the slope of the tangent to the equilibrium path approaches zero. The ideal solution would place the critical-limit load at the point where the slope goes to zero and all of the member strain energy densities become equal.

A recurrence relation is used to update the design variables in the Hrinda optimization algorithm--in this case, the cross-sectional areas of the truss members. Recurrence equations are used in iterative processes to keep track of and adjust variables in optimizations. Formulation of the updating recurrence relation relies on information from the arc length parameters. One of those parameters is the incremental external-load multiplier. The multiplier that is selected to update the recurrence equation depends on which phase the arc length solver is being processed. The solver has three phases; each phase checks the difference between the present total external load multiplier and the next load multiplier. This process is shown in the Hrinda optimization algorithm flowchart of figure 31. The checks indicate how close the load is to reaching a limit point and whether the arc length is becoming more horizontal with a decreasing slope.

7.1 Scale Strain Parameter

A new parameter used in the Hrinda optimization algorithm, called the scale strain parameter, is developed to use the size of the incremental external load at the end of each arc length iteration. The value of the scale strain parameter depends on the recurrence checks that are performed (see figure 30). The three possible values of the scale strain parameter *SSP* are

$$SSP = \lambda_0 + \Delta\lambda_0,\ \lambda_i + \Delta\lambda_i, \text{ or } \lambda_k + \Delta\lambda_k \tag{7.1}$$

where $\Delta\lambda_0$, $\Delta\lambda_i$ and $\Delta\lambda_k$ are the incremental load multipliers that are used along the load path. The initial load increment is $\Delta\lambda_0 = dl0$ and is supplied by the user at the start of the arc length procedure, as shown in figure 5. The value λ_0 is a preload value that is placed on the structure; this value is zero for the problems shown in this document. Figure 30 shows the other two incremental multipliers as $\Delta\lambda_i = dli$, which is the incremental external multiplier at step i and $\Delta\lambda_k = dlk$, which is the incremental external multiplier at step k.

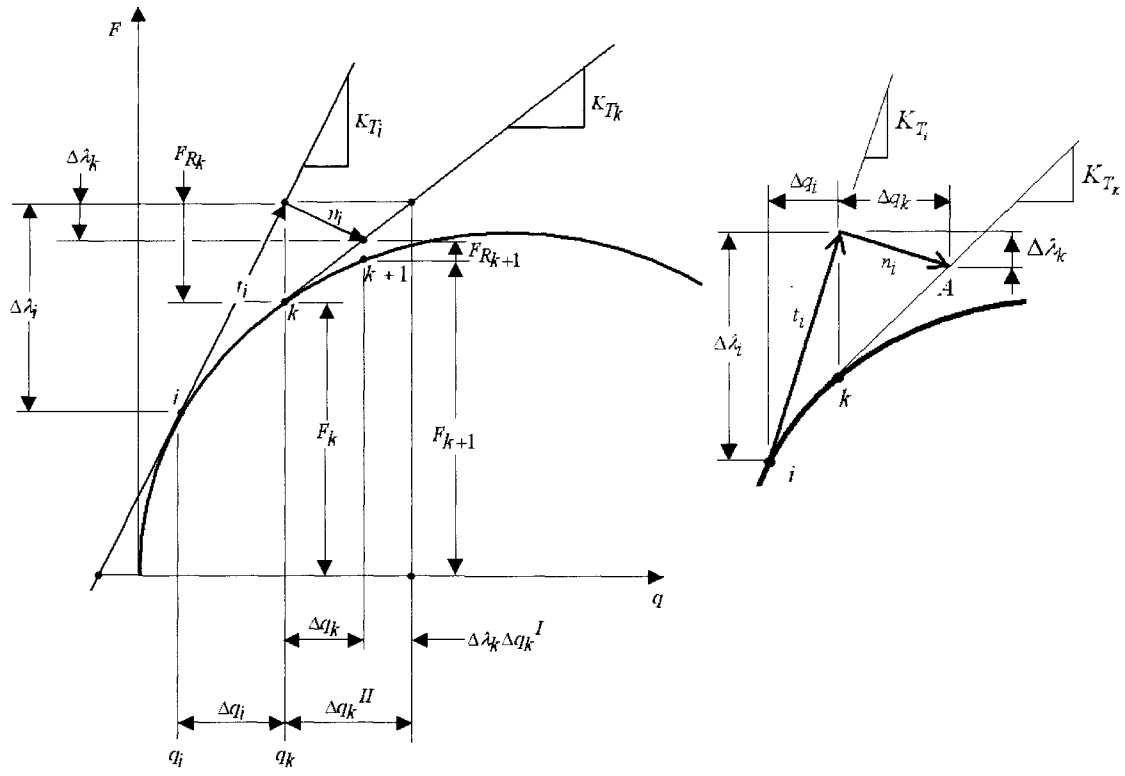


Figure 30. The Riks-Wempner arc length method on a normal plane for a single-degree-of-freedom system.

The new recurrence formula for scaling the design variable is taken from equation (6.50) and can be written by using the *SSP* as

$$\text{SEDscale} = (\text{SEDtotal}/\text{SEDtotal2})[\text{SED(i)}]^{\text{SSP}} \tag{7.2}$$

where

SEDscale is the strain energy density scaling factor (7.3)

SEDtotal is the total strain energy density (7.4)

SEDtotal2 is SEDtotal^2 (7.5)

SED(i) is the strain energy density in the *i*th member (7.6)

SSP is the new scale strain parameter (7.7)

dSEDtol = user-given differences between the strain energy densities for all elements (7.8)

The flowchart that is shown in figure 31 outlines the scheme for updating the area design variable *propsect*. Immediately after a solution is found in the arc length solver, the elemental strain energies are calculated and compared with the strain energy density tolerance *dSEDtol*. If all members are within the tolerance, then the solution is optimal. If one member fails the tolerance test then a new *SSP* is calculated from the initial multiplier *dl0* or the incremental external multipliers *dli* and *dlk*.

The strain energy density check is shown in the flowchart in figure 31 and graphically explained in figure 30. Each user-defined tolerance depends on the type of model, the level of fidelity that the user selects for in weight reduction, and how the structure was initially modeled. The whole structure can be optimized for weight, or truss members can be grouped together for sizing. If the absolute difference in strain energy density is below the tolerance, then the solution is considered to be at its optimum. A minimum area is used to avoid zero- area trusses. A zero-area indicates that the truss member is necessary for a stable structure but carries no load. The minimum-area check is made to ensure that member sizes are greater than the user-provided area.

Another check of all of the strain energy densities is made using the user-provided *dSEDtol*. Only solid rods are used in this work, but any section could be used. The new strain energy densities are then computed with the updated areas.

The strategies that are explained for using the recurrence formula will be demonstrated using sample problems that are solved using a nonlinear solution in NASTRAN. The results of other nonlinear structures that have been published in

numerous references will also be used to verify the proposed algorithm with the strain energy density and arc length procedure.

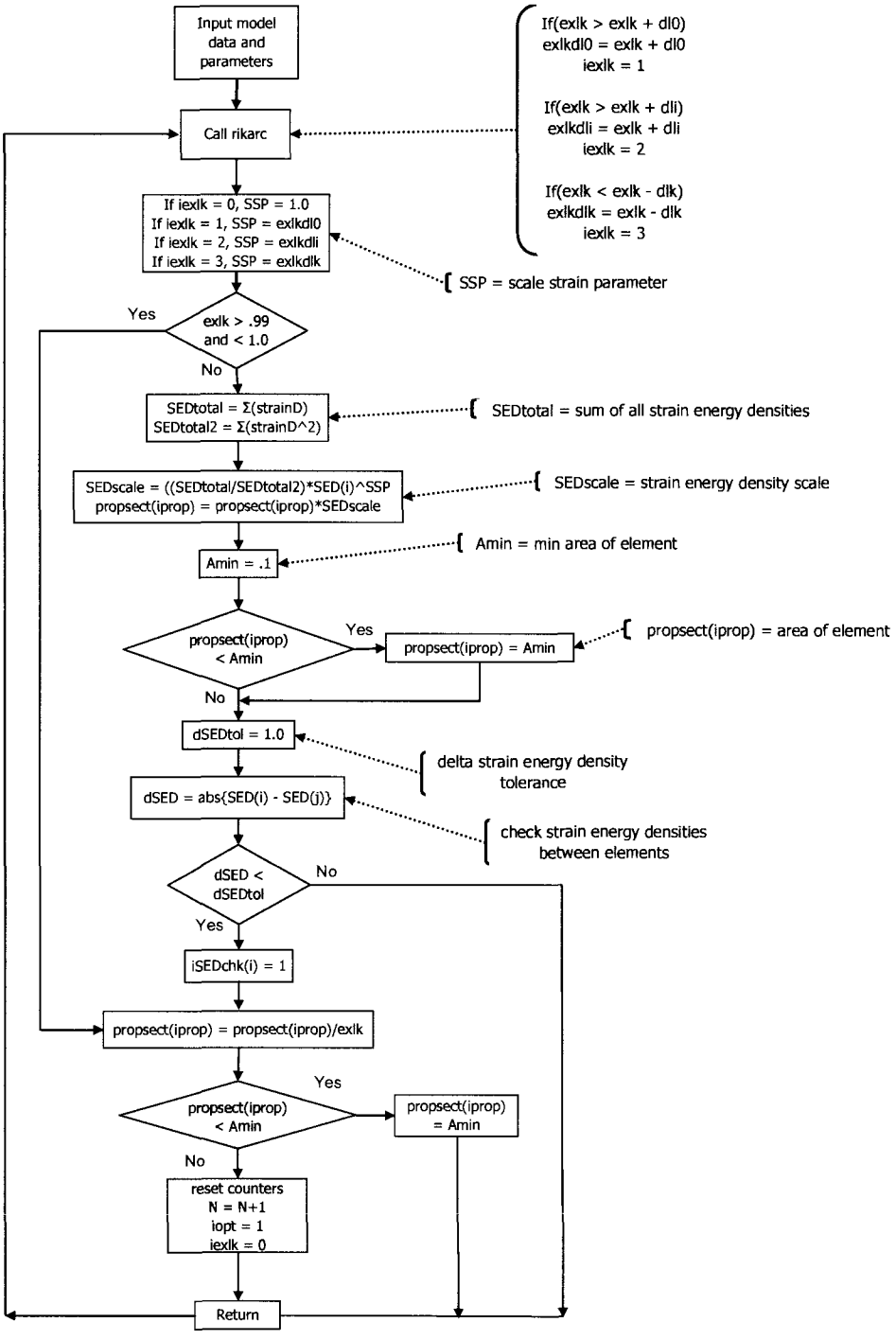


Figure 31. Flowchart for the Hrinda optimizer program.

7.2 Program Outline

The finite element analysis is implemented in the FORTRAN subroutine "rikarc." This main routine assembles and solves the finite element model using an arc length method to trace the geometrically nonlinear equilibrium path. The "rikarc" subroutine also contains other routines that use sparse storage and sparse equation solving techniques. A listing of the program may be found in appendix C.

Within "rikarc" is the major subroutine "syseqn" that assembles the nonlinear system of equations from global stiffness matrix that is stored in a one-dimensional array. Inside "syseqn" are numerous other subroutines that apply portions of a sparse equation solution algorithm developed by Nguyen [19]. The subroutine "symbass" starts the process to assemble a symmetrical sparse stiffness matrix by assigning pointers to the first non-zero row term of the element connectivity.

The main FORTRAN subroutines are listed below in the order they are called in the algorithm and explained in the following paragraphs.

```
call generalip
call materprop
call sectprop
call nodecoor
call elconect
call loads
call supportdof
```

Call Rikarc

```
    call syseqn
        call symbass
        call copy_int
        call assembly
            call info_elstif
            call elstif
            call numass
```

```
        call minfills
        call symfactd
        call copy_int
        call transad
            call transa2d
        call supernode
        call numfa1d
        call fbed

    call resid
    call test
    call result
    call itnrml
        call syseqn
        call resid
        call test
```

The first block of subroutines, "generalip" through "supportdof," are used to input general information about the finite element model. Subroutine "generalip" will read an input file containing: number of supports, number of degree-of-freedom at each node, the degree-of-freedom of each load, number of material properties, number of section properties and the type of element. Subroutine "materprop" reads in the material properties and "sectprop" reads in the cross sectional properties of all the finite elements. The nodal xyz coordinates are read by the subroutine "nodecoor." Input of the element connectivity is accomplished by the subroutine "elconect" which reads the order and element number from the input data and places them into a one-dimensional array, "ie". The subroutine "loads" reads all the applied loads given in the input file. The subroutine "supportdof" also reads the support degree-of-freedoms from the input file.

CHAPTER VIII

VERIFICATION EXAMPLES OF OPTIMIZED TRUSSES

The following sample problems are used to verify the proposed nonlinear optimization method. All of the examples attempt to find the minimum weight of truss areas to support an applied load without buckling. The problems are shallow trusses that have critical limit points and are taken from Khot and Kamat [13] and reviewed in [26]. The first two problems solve a two-element, two-dimensional (2-D) system. A vertical load is applied at the mid-point with pinned end conditions. The next problem is a four-element system that has four different lengths with four cross sections to be optimized. Then, a larger star dome problem is presented, which is similar to the Crisfield star dome in volume two [11]. This problem was minimized by using four cross-sectional area groups that were related to the layout of the design. The last problem is a large shallow truss that is more correct to space trusses. The problem allows for each of the members to be solved for a unique design variable. Here, the system has 13 cross sections that are sized to support a vertical load without snapping through. The solutions that were found in all the examples were compared with those solutions that were published by Khot, as well as with NASTRAN. The areas found by the proposed technique were used in the NASTRAN models. Here, the displacements of the NASTRAN solutions were checked against those found by the proposed algorithm. The first example results are also compared with an analytical solution given by Crisfield [10].

8.1 Two-Member Symmetric Truss Example

The first example that is used to verify the proposed algorithm is a two-member symmetric shallow truss with a 200-lb concentrated load, as shown in figure 32. The problem is taken from Khot and Kamat [13] and is optimized to find the minimum sectional area that can support the 200-lb load without a sudden strain energy release and structure snap-through. All degrees-of-freedom were constrained except at the center node.

Two different initial areas of $3.0\,in^2$ and $5.0\,in^2$ were chosen to test the ability of the algorithms to find a unique area for each member. Table 3 shows the initial and final strain energy density distributions in the two elements. The strain energy density in the members become nearly equal at 0.878, which makes the final relative strain energy density equal to 1.0 to indicate an optimal design. Both members were sized to equal areas of $6.499\,in^2$, slightly larger than the results that were found by Khot and Kamat in [13] (i.e. $6.4977\,in^2$ for both members). The total weight summary of the members after each iteration is given in table 4.

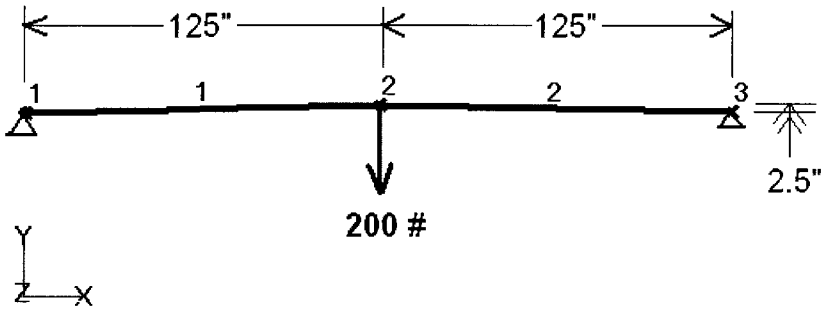


Figure 32. Two-element symmetric truss with 200-lb apex load.

Table 3. Strain energy density distribution for two-element symmetric truss.

Element	Initial Area, in^2	Initial SED	Initial Relative SED	Final SED	Final Relative SED	Final Area, in^2
1	3.00	1.38806	1.00000	0.87799	1.00000	6.4990
2	5.00	0.49960	0.35993	0.87802	1.00000	6.4989

Table 4. Iteration weight history for two-element symmetric truss.

Iteration	Weight, lb	exlk
1	0.100020E+03	0.577049E+00
2	0.803310E+02	0.493552E+00
3	0.799019E+02	0.491683E+00
3	0.799019E+02	0.491682E+00
4	0.162507E+03	0.100000E+01

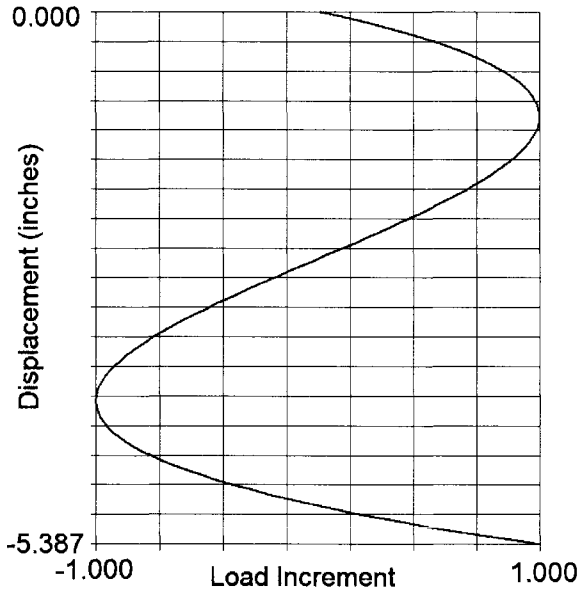


Figure 33. Khot and Kamat load displacement curve for two-member symmetric truss with cross-section areas equal to 6.4977 in^2.

The published results given in [13] used an area of 6.4977 in^2 and are close to optimal. However, the NASTRAN and analytical results as well as the Hrinda solution show the truss will not support the 200-lb load; snap-through buckling will still occur. The smaller area (i.e., 6.4977 in^2 from [13] was used in a NASTRAN analysis to obtain the results that are plotted in figure 33. The equilibrium path shows two critical-limit points, with the first at a load increment of .99959, or an applied load of 200 lb times .99959, which equals 199.9180 lb. The first limit point is reached, and then the structure responds with a snap-through before reaching the intended design load. An analytical solution was also obtained using the exact load displacement path equation given in Crisfield [10]. Here the analytical result also showed that a slightly larger area of 6.499 in^2 was required to support the 200-lb load.

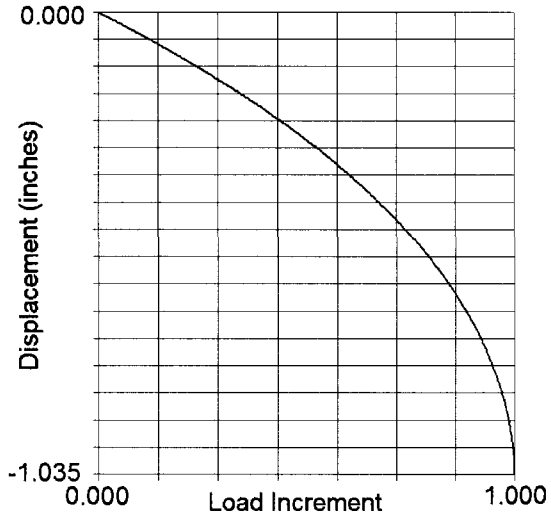


Figure 34. Hrinda load displacement curve for two-member symmetric truss with cross-section areas equal to 6.499 in^2.

The areas obtained from the proposed algorithm were substituted into the NASTRAN model of the two-element symmetric truss that was used in figure 34. The areas for both elements were the same as can be expected from symmetry (i.e., $6.499\,\text{in}^2$). This optimal cross section produces a maximum displacement of 1.0354 in. at the maximum load increment of 1.0, as shown in figure 34. The area obtained from the proposed Hrinda algorithm was slightly higher than in [13] and did not cause an instability snap-through. The differences between the Hrinda and Khot/Kamat results are negligible. The two results demonstrate the sensitivity of the optimal solution to the number of significant digits used in the design and solution parameters. If a larger load increment is used during the arc length solution, the design area may be slightly smaller as in [13]. The two designs are nearly identical but illustrate how a slight imperfection can affect the design, as stated by Bruns et al. [27], Khot and Kamat [13] and Sedaghati et al. [18]. Appendix B.1 lists the computer model file and output results used in creating the equilibrium path in figure 34.

8.2 Two-Member Asymmetric Truss Example

The next verification problem is an asymmetric, two-element truss, shown in figure 35. The two-element truss has the same 200-lb applied load and support conditions as in the first example. The problem is also taken from Khot and Kamat [13]. A summary of the strain element density results that were obtained with the proposed algorithm is given in table 5. The optimal area is achieved after the difference between the strain energy density in both members falls within the user-defined parameter *dSEDtol*. The final optimal areas were $2.6673\,\text{in}^2$ and $2.6641\,\text{in}^2$; for a practical design, they were the same in both members. The weight history is shown in table 6 for each iteration, with a final

design weight of 66.64 lb. This result compares closely with the result from Khot and Kamat of 66.53 lb [13]. Appendix B.2 gives the computer model input file used in producing the results listed in tables 5 and 6.

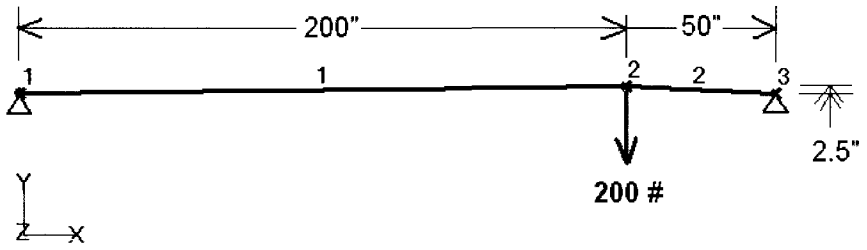


Figure 35. Two-element asymmetric truss.

Table 5. Area and strain energy density distribution of two-element asymmetric truss.

Element	Initial Area, in^2	Initial SED	Initial Relative SED	Final Area, in^2	Final SED	Final Relative SED
1	1.00	1.5047	0.2497	0.266730E+01	2.1532	1.0000
2	2.00	6.0266	1.0000	0.266409E+01	2.1495	0.9983

Table 6. Iteration weight history of two-element asymmetric truss.

Iteration	Weight, (lb)	exlk
1	0.450094E+02	0.625349E+00
2	0.241347E+02	0.360397E+00
3	0.246053E+02	0.369088E+00
4	0.248345E+02	0.372661E+00
5	0.249026E+02	0.373692E+00
6	0.249205E+02	0.373962E+00
7	0.666391E+02	0.100000E+01

8.3 Four-Member Asymmetric Truss Example

The next problem that was investigated was an optimized, asymmetric four-element truss with stability constraints that were presented by Khot and Kamat [13] and analyzed using the Hrinda optimization approach. The model is shown in figures 36 and 37 with a vertical loading of 200 lb at the center node. The first part of the analysis was to create a nonlinear NASTRAN model and test the results that were given in [13] for possible snap-through behavior. The areas that were used in the NASTRAN analysis were the final design areas from [13]; these are shown in table 7. The load deflection plot of the center node is displayed in figure 38; the plot shows that a critical-limit point is reached before the design load of 200 lb. Table 9 lists the four main data points shown by the plot in figure 38. A critical-limit point is reached at a load increment of 0.9174 with a displacement equal to 1.2597 in. Strain energy that is stored in the structure is released, and the load decreases with increasing load. The load goes to zero at about 3.0 in. (point B in fig. 38) and changes direction until another limit point is reached at a maximum load increment of 0.9174 at a displacement equal to 4.7217 in. (point C in figure 38). The curve shown in figure 38 demonstrates that the structure is only able to support a load of 0.9174×200 lb or 183.48 lb. Figure 37 is a plot of the deflected shape after snap-through.

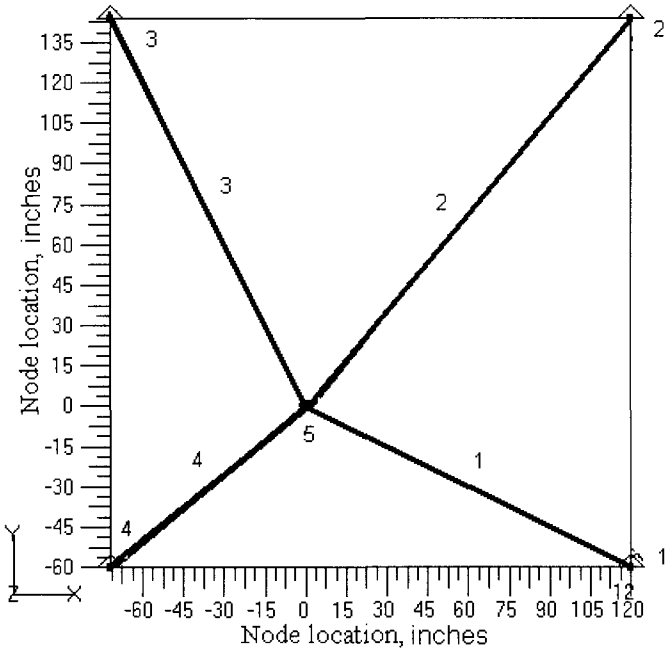


Figure 36. Plan view of the four-member asymmetric truss example.

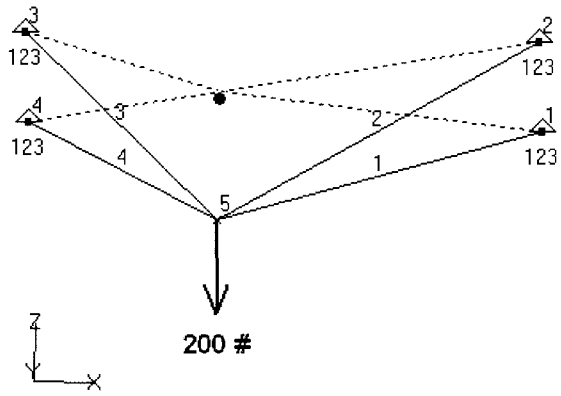


Figure 37. Four-member truss at snap- through.

Table 7. Khot [13] element properties.

Element	Area, in^2
1	0.217240E+01
2	0.163832E+01
3	0.172547E+01
4	0.292330E+01

Now the Hrinda optimization algorithm method is used to solve and optimize the same problem that was given in [13]. The initial area for all of the members was $2.20\,in^2$, with the final areas shown in table 8. The final optimized weight is 115.066 lb, as shown in the iteration history that is given in table 10. The final results that were derived from the Hrinda method were used in a NASTRAN nonlinear analysis; the results for the center node are plotted in figure 39.

Table 8. Hrinda final design.

Element	Area, in^2
1	0.216836E+01
2	0.165063E+01
3	0.173753E+01
4	0.288408E+01

The load increment in figure 39 goes to 1.0 with a displacement of 1.249 in. This plot demonstrates that the structure can support the 200-lb load without snap-through buckling. The maximum load increment is reached at 1.0 and is multiplied by the applied load of 200 lb. The strain energy density distribution results for each iteration found by Hrinda are summarized in table 11. The results show that the strain energy densities in all of the members are nearly equal in iteration 20.

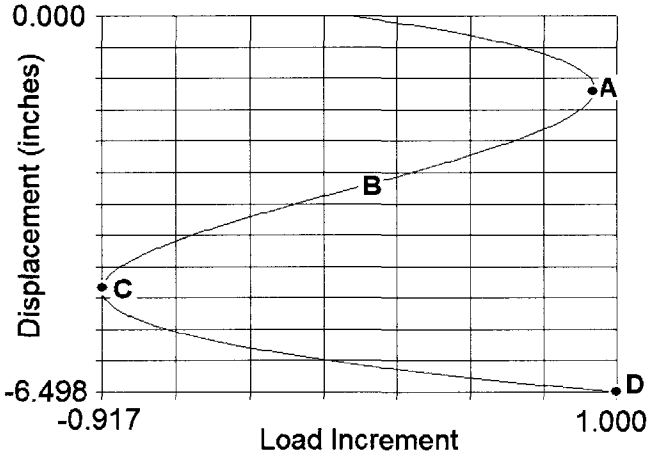


Figure 38. Four-member asymmetric truss equilibrium path during snap-through.

Table 9 lists key quantitative results that are indicated in figure 38.

Table 9. Quantitative results of four-member truss problem using NASTRAN that are shown in figure 38 given as (load increment, displacement).

A	B	C	D
.9174, 1.2597	0.000, 3.00	-.9174, 4.7217	1.000, 6.498

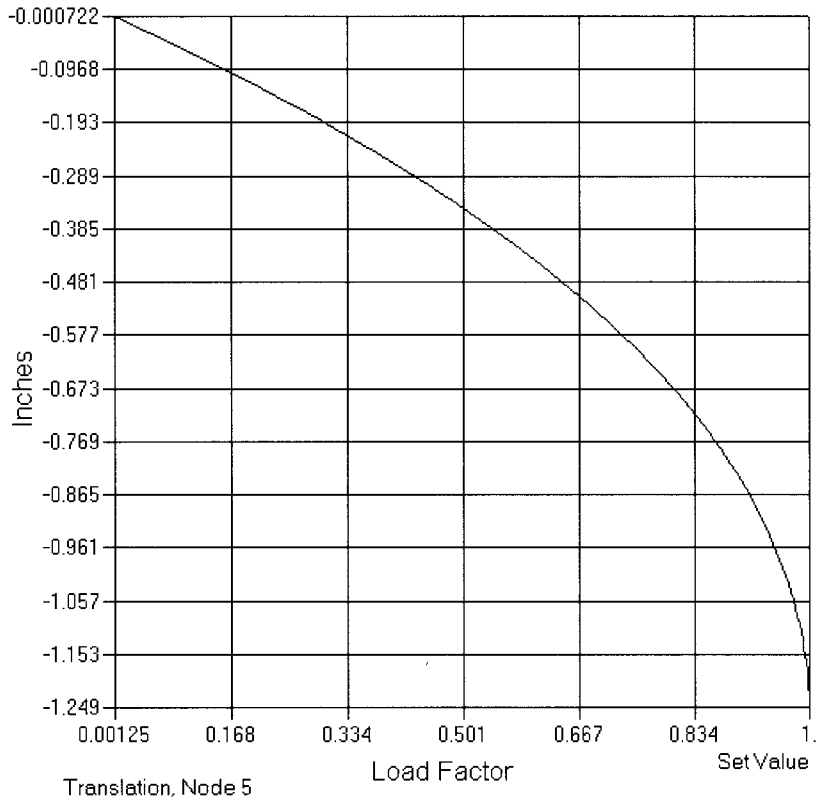


Figure 39. Load increment-displacement results for center node of four-member truss using Hrinda results in NASTRAN.

Table 10. Iteration weight history for four-member truss.

Iteration	Weight, lb.	exlk
1	0.12682196E+03	0.100030E+01
2	0.12177516E+03	0.100013E+01
3	0.11858018E+03	0.100004E+01
4	0.11654891E+03	0.100001E+01
5	0.11525130E+03	0.993698E+00
6	0.10874612E+03	0.937731E+00
7	0.10487218E+03	0.905957E+00
8	0.10064500E+03	0.871200E+00
9	0.98937661E+02	0.858015E+00
10	0.97439280E+02	0.845865E+00
11	0.97044707E+02	0.842990E+00
12	0.96626275E+02	0.839536E+00
13	0.96581425E+02	0.839265E+00
14	0.96460005E+02	0.838257E+00
15	0.96471788E+02	0.838386E+00
16	0.96431562E+02	0.838045E+00
17	0.96443952E+02	0.838159E+00
18	0.96428782E+02	0.838028E+00
19	0.96435934E+02	0.838092E+00
20	0.11506603E+03	0.100000E+01

Table 10 shows the weight and load factors for each iteration during the Hrinda analysis. The first iteration has the load level at 1.00 and the weight at 126.82 lbs. This means that the load was fully applied, but the element SED differences were too large and exceeded the SED tolerance of dSEDtol=1.d-2.

Table 12 summarizes the Hrinda results by listing the initial SED values and comparing them when an optimal solution is reached. The last two columns show how the final SED values are nearly 1.0, thus indicating the optimal design has been found.

Table 11. SED per element at each iteration from Hrinda results.

Iteration No.	Element	SED
1	1	1.054980E+00
	2	6.206980E-01
	3	6.735780E-01
	4	1.890250E+00
2	1	1.188370E+00
	2	7.835400E-01
	3	8.288730E-01
	4	1.902710E+00
3	1	1.321840E+00
	2	9.517260E-01
	3	9.900690E-01
	4	1.931050E+00
4	1	1.526660E+00
	2	1.177230E+00
	3	1.210970E+00
	4	2.070520E+00
5	1	1.560360E+00
	2	1.269670E+00
	3	1.296040E+00
	4	1.992890E+00
6	1	1.438140E+00
	2	1.743000E+00
	3	1.731370E+00
	4	1.122350E+00
7	1	1.553770E+00
	2	1.300650E+00
	3	1.322680E+00
	4	1.921000E+00
8	1	1.459330E+00
	2	1.675270E+00
	3	1.665410E+00
	4	1.223960E+00
9	1	1.536430E+00
	2	1.381530E+00
	3	1.393620E+00
	4	1.747390E+00
10	1	1.482270E+00
	2	1.596440E+00
	3	1.590250E+00
	4	1.350300E+00
11	1	1.521010E+00
	2	1.443440E+00
	3	1.448930E+00
	4	1.621580E+00
12	1	1.495660E+00
	2	1.549700E+00
	3	1.546510E+00
	4	1.431000E+00
13	1	1.512990E+00
	2	1.476600E+00
	3	1.479020E+00
	4	1.558940E+00
14	1	1.501880E+00
	2	1.526780E+00
	3	1.525260E+00
	4	1.471600E+00
15	1	1.510310E+00
	2	1.493510E+00
	3	1.494590E+00
	4	1.531240E+00
16	1	1.503250E+00
	2	1.514660E+00
	3	1.513950E+00
	4	1.489290E+00
17	1	1.508780E+00
	2	1.501070E+00
	3	1.501550E+00
	4	1.518320E+00
18	1	1.504550E+00
	2	1.509780E+00
	3	1.509460E+00
	4	1.498140E+00
19	1	1.506280E+00
	2	1.502740E+00
	3	1.502960E+00
	4	1.510630E+00
20	1	1.491650E+00
	2	1.488160E+00
	3	1.488370E+00
	4	1.495930E+00

Table 12. Area and strain energy density distribution in four-member truss.

Element	Initial Area, in^2	Initial SED	Initial Relative SED	Final Area, in^2	Final SED	Final Relative SED
1	2.20	0.105498E+01	0.558117	0.216836E+01	0.149165E+01	.997139
2	2.20	0.620698E+00	0.328368	0.165063E+01	0.148816E+01	.994806
3	2.20	0.673578E+00	0.356343	0.173753E+01	0.148837E+01	.994946
4	2.20	0.189025E+01	1.000000	0.288408E+01	0.149593E+01	1.00000

Table 13. Effects of "dSEDtol" value on final weight and number of iterations.

dSEDtol	Final Weight #	Iterations
2.0	0.12682196E+03	1
1.0	0.11858018E+03	3
0.8	0.11525130E+03	5
0.5	0.11552458E+03	9
0.1	0.11507860E+03	14
0.01	0.11506603E+03	20

The effect on the final weight of using different values for "dSEDtol" is shown in table 13. Large values of this parameter were sized for a heavier structure but with less iterations. As the parameter value was reduced, the weight of the structure approached the optimum but more iterations were required. This shows how the parameter may be used when analyzing a new structure. The value could be set to a larger value and then gradually made smaller until a final weight value stabilizes, thus indicating the optimal weight is being approached. If decreasing the "dSEDtol" parameter has little effect on the final weight, the user may also increase the value to reduce the number of iterations.

The element SEDs at the end of each iteration are listed in table 11. The SED in the first iteration for elements 1-4 are: 1.05498, 0.620698, 0.673578 and 1.89025, respectively. The SEDs of the elements are not equal to their differences above the SED

tolerance of 1.d-2. The program adjusts the element areas for the next iterations until the last iteration is reached. The load at the last iteration is fully applied with the element SEDs for elements 1-4 as: 1.49165, 1.48816, 1.48837 and 1.49593, respectively.

The number of iterations may be reduced by raising the element SED tolerance values. A large "dSEDtol" value causes the program to stop at a weight slightly heavier. Another run of the program used a "dSEDtol=.1" to show the result of raising the tolerance value. Table 14 shows the effect of using different initial areas as the final weight and the number of iterations to reach a solution. The results show that initial values affect solution times. The table also lists the sum of the initial area, $A_{initial}$, divided by the final area, A_{final} , for the four elements. The larger the sum of the ratios becomes, the more iterations are required to reach a solution.

Table 14. Effect of using different initial areas on the final weight and number of iterations required to each a final design. All elements used the same initial area. The difference in strain energy density, dSEDtol, was equal to .001 for the five runs in this table.

Initial Areas Elements 1-4	Final Weight #	Number of Iterations	$\sum \frac{A_{initial}}{A_{final}}$
1.00	0.11505859E+03	5	3.1492
2.00	0.11505639E+03	7	3.9787
2.20	0.11506603E+03	20	4.3764
3.00	0.15386057E+03	19	5.9678
4.00	0.20518989E+03	16	7.9572

8.4 Star Dome Truss Example

The next problem that is presented is the shallow star dome truss example that is introduced by Khot and Kamat [13] and shown in figure 40. The design has 30 members with pin supports at the outer nodes and one vertical load at the center. The design sizing is formed on four optimization groups, which are listed in table 15. The minimum area for all elements is 0.1 in^2. A downward vertical load of 200 lbs is applied at the center of the structure.

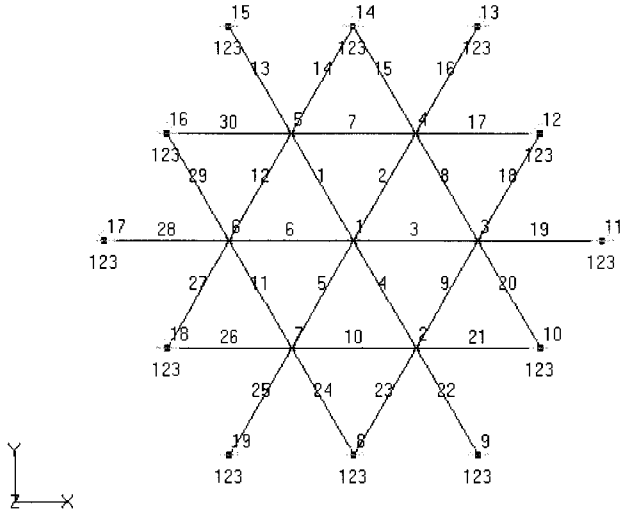


Table 15. Star dome optimization groups.

Group I	1 - 6
Group II	7 - 12
Group III	13, 16, 19, 22, 25, 28
Group IV	14, 15, 17, 18, 20, 21, 23, 24, 26, 27, 29, 30

Figure 40. Star dome truss example.

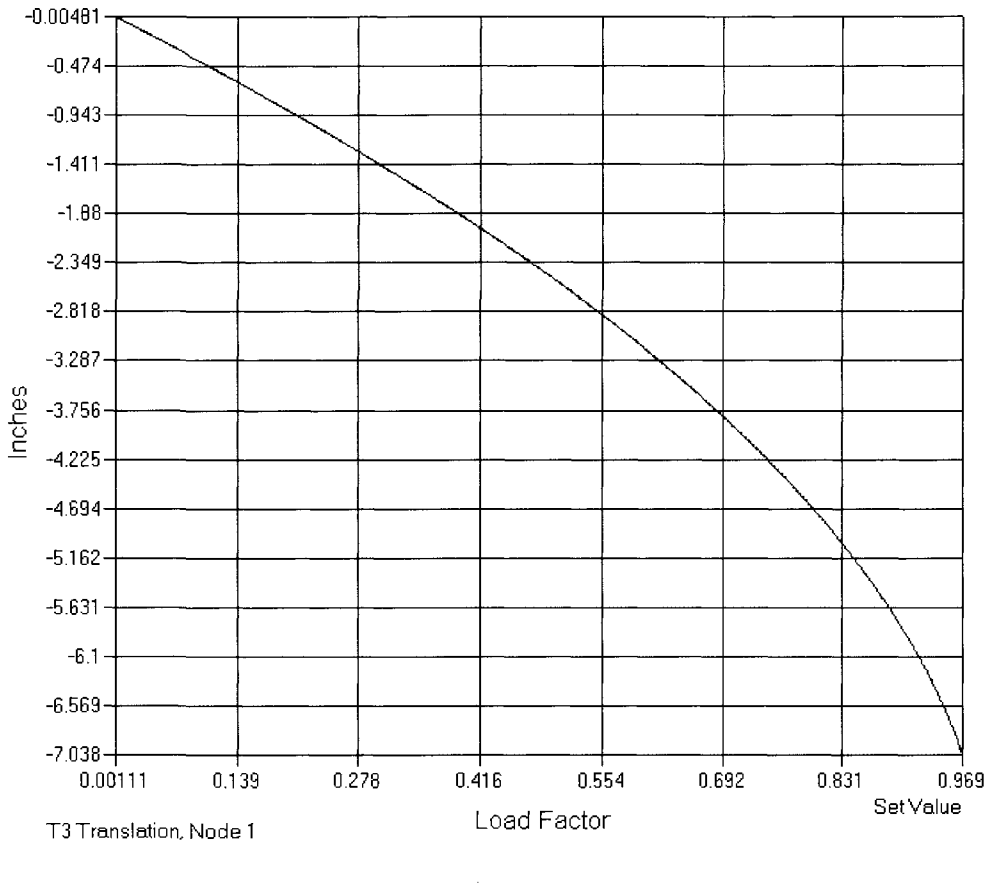


Figure 41. Load-displacement results for center node of star dome truss using Khot results in NASTRAN.

Table 16. Star dome strain energy density and truss area design.

Element	Initial SED	Initial Relative SED	Final SED	Final Relative SED	Final Area, in^2
1	0.194286E+02	1.000000	0.156044E+02	1.000000	0.168767E+01
2	0.194286E+02	1.000000	0.156044E+02	1.000000	0.168767E+01
3	0.194286E+02	1.000000	0.156044E+02	1.000000	0.168767E+01
4	0.194286E+02	1.000000	0.156044E+02	1.000000	0.168767E+01
5	0.194286E+02	1.000000	0.156044E+02	1.000000	0.168767E+01
6	0.194286E+02	1.000000	0.156044E+02	1.000000	0.168767E+01
7	0.126056E+02	0.648817	0.156121E+02	0.999507	0.136980E+01
8	0.126056E+02	0.648817	0.156121E+02	0.999507	0.136980E+01
9	0.126056E+02	0.648817	0.156121E+02	0.999507	0.136980E+01
10	0.126056E+02	0.648817	0.156121E+02	0.999507	0.136980E+01
11	0.126056E+02	0.648817	0.156121E+02	0.999507	0.136980E+01
12	0.126056E+02	0.648817	0.156121E+02	0.999507	0.136980E+01
13	0.740568E+00	0.038117	0.160748E+02	0.970737	0.262472E+00
14	0.183099E-05	0.000000	0.428412E+01	0.274546	0.100000E+00
15	0.183099E-05	0.000000	0.428412E+01	0.274546	0.100000E+00
16	0.740568E+00	0.038117	0.160748E+02	0.970737	0.262472E+00
17	0.183208E-05	0.000000	0.428411E+01	0.274546	0.100000E+00
18	0.183349E-05	0.000000	0.428413E+01	0.274546	0.100000E+00
19	0.740566E+00	0.038117	0.160748E+02	0.970737	0.262472E+00
20	0.183349E-05	0.000000	0.428413E+01	0.274546	0.100000E+00
21	0.183208E-05	0.000000	0.428411E+01	0.274546	0.100000E+00
22	0.740568E+00	0.038117	0.160748E+02	0.970737	0.262472E+00
23	0.183099E-05	0.000000	0.428412E+01	0.274546	0.100000E+00
24	0.183099E-05	0.000000	0.428412E+01	0.274546	0.100000E+00
25	0.740568E+00	0.038117	0.160748E+02	0.970737	0.262472E+00
26	0.183208E-05	0.000000	0.428411E+01	0.274546	0.100000E+00
27	0.183349E-05	0.000000	0.428413E+01	0.274546	0.100000E+00
28	0.740566E+00	0.038117	0.160748E+02	0.970737	0.262472E+00
29	0.183349E-05	0.000000	0.428413E+01	0.274546	0.100000E+00
30	0.183208E-05	0.000000	0.428411E+01	0.274546	0.100000E+00

Table 17. Star dome weight summary.

Iteration	Weight #
1	0.108690E+04
2	0.496329E+03
3	0.516501E+03
4	0.551977E+03
5	0.591325E+03
6	0.638388E+03
7	0.694186E+03
8 (final)	**0.762154E+03**

Table 18. Final optimized group truss areas.

Group	Area, in^2 Hrinda	Area, in^2 Khot/Kamat
I	0.168767E+01	1.6926
II	0.136980E+01	1.3754
III	0.262472E+00	0.2693
IV	0.100000E+00	0.1000

The strain energy density summary and the final member areas are given in table 16. The relative strain energy densities are also given; group IV is sized to the minimum $0.10\,in^2$. Group IV members have been sized to the minimum area so that their relative strain energy densities do not approach unity. Table 17 shows the total design weights after each iteration, with a final weight of 762.154 lb. This weight compares well with the final optimized design weight of 766.188 lb given by Khot and Kamat [13]. The group sizing results are shown in table 18; these results also compare closely with the results from Khot and Kamat [13].

The Khot areas were used in a NASTRAN model with the results for the center node equilibrium path plotted in figure 41. The NASTRAN analysis ended with an error stating that the solution ended prematurely and the analysis could not continue to the full loading.

8.5 Large Shallow Truss Example

The next problem is a large symmetric shallow truss, which is presented by Khot and Kamat in [13]. The design (see figure 42) includes 23 members that are pinned at node 1 and sized for two applied loads of 300 lb and 600 lb with symmetric boundary conditions at the apex.

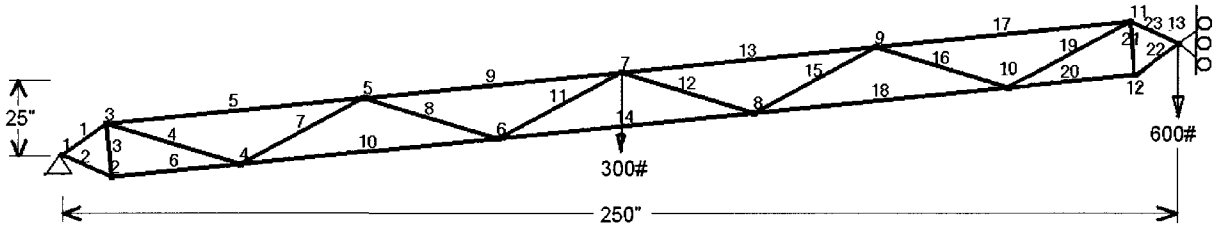


Figure 42. Shallow truss half-symmetry model with two point loads.

Table 19. Shallow truss strain energy density and final design areas.

Element	Initial SED	Initial Relative SED	Final SED	Final Relative SED
1	0.265608E+01	0.81200	0.159399E+01	0.99744
2	0.230073E+01	0.70337	0.159181E+01	0.99608
3	0.574825E+00	0.17573	0.156642E+01	0.98019
4	0.209752E-01	0.00641	0.150678E+01	0.94287
5	0.238759E+01	0.72992	0.159198E+01	0.99618
6	0.172523E+01	0.52743	0.158655E+01	0.99279
7	0.207116E-01	0.00633	0.150819E+01	0.94375
8	0.199578E-01	0.00610	0.150978E+01	0.94475
9	0.327102E+01	1.00000	0.159808E+01	1.00000
10	0.109587E+01	0.33502	0.157836E+01	0.98766
11	0.193169E-01	0.00591	0.150947E+01	0.94455
12	0.192245E-01	0.00588	0.150930E+01	0.94445
13	0.323500E+01	0.98899	0.159787E+01	0.99987
14	0.621099E+00	0.18988	0.156713E+01	0.98063
15	0.196568E-01	0.00601	0.150949E+01	0.94456
16	0.201948E-01	0.00617	0.150776E+01	0.94348
17	0.236487E+01	0.72298	0.159179E+01	0.99606
18	0.109313E+01	0.33419	0.157830E+01	0.98762
19	0.206514E-01	0.00631	0.150652E+01	0.94271
20	0.171458E+01	0.52417	0.158643E+01	0.99271
21	0.564183E+00	0.17248	0.156607E+01	0.99221
22	0.227937E+01	0.69684	0.159163E+01	0.99596
23	0.262691E+01	0.80309	0.159378E+01	0.99731

Table 20. Hrinda and Khot Area comparisons.

Element	Final Area, in^2 (Hrinda)	Final Area, in^2 (Khot)
1	0.257384E+01	2.4747
2	0.240403E+01	2.3695
3	0.121047E+01	1.1837
4	0.231500E+00	0.2609
5	0.243846E+01	2.3694
6	0.208529E+01	2.0551
7	0.231386E+00	0.2259
8	0.229998E+00	0.2227
9	0.284772E+01	2.7512
10	0.167302E+01	1.6353
11	0.227155E+00	0.1945
12	0.226557E+00	0.1941
13	0.283215E+01	2.7463
14	0.126479E+01	1.2751
15	0.228240E+00	0.2230
16	0.228500E+00	0.2249
17	0.242697E+01	2.3649
18	0.167087E+01	1.6351
19	0.229750E+00	0.2590
20	0.207890E+01	0.0524
21	0.119933E+01	1.1756
22	0.239295E+01	2.3626
23	0.255980E+01	2.4687

Table 21. Hrinda shallow truss weight results.

Iteration	Weight, #
1	0.156524E+03
2	0.134545E+03
3	0.123092E+03
4	0.116857E+03
5	0.113505E+03
6	0.111891E+03
7	0.111382E+03
8	0.111581E+03
9	0.112211E+03
10	0.113076E+03
11	0.114034E+03
12	0.114992E+03
13	0.115892E+03
14	0.116704E+03
15	0.117415E+03
16	0.118024E+03
17	0.118539E+03
18	0.118968E+03
19	0.119322E+03
20	0.119612E+03
21 (final)	**0.119850E+03**

The strain energy density summary is listed in table 19, and the final member areas are shown in table 20. A *dSEDtol* = .10 is imposed for all elements. The member areas that are derived with the Hrinda optimization algorithm are listed next to those of Khot and Kamat [13] for comparison in table 20. All of the members are in close agreement except member 20. This is because the Hrinda results imposed a minimum area of 0.1 in^2 so that the optimizer was forced to size accordingly. The Hrinda weight summary is given in table 21 with the final weight of 119.85 lb. Doubling this value to account for symmetry yields the final shallow truss weight of 239.70 lb, which is close to the final weight of 234.13 lb found by Khot and Kamat [13].

CHAPTER IX

DISCUSSION

The Lagrange multiplier (γ) is written as a function of the strain energy densities (SED) in equation (6.49). The equation is placed into the design variable equation (6.45) to form a simple updating equation (6.50). The exponential recurrence formula (6.50) out-performed other updating relationships by being able to adjust the design variable to obtain the correct minimum weights given by the chapter 8 examples. Other relationships were attempted; however, they could not achieve a uniform SED. An optimal structure exists when all members have the same SED. This optimality criteria was introduced in equation (6.43) which is repeated below

$$\gamma_1 \left(SED_i \right) = 1 \tag{9.1}$$

According to this relationship, when all the member SED s in a structure are the same an optimal design exists. A practical design will not have an exact adherence to equation (9.1) but allows the SED_i s in the members to approach uniformity. This was observed in the example problems of chapter 8 where the SED s are shown to approach uniform values. More iteration would be required to reduce the residuals between the member SED s in equation (6.46).

Other design variable updating relationships failed most likely from not using statistical modeling of the residuals as discussed in chapter six. Minimizing the Lagrange multiplier in the sum of the residual equation (6.45) produced acceptable results. Other statistical modeling techniques for measuring the difference in member's SED s could be incorporated into the program to investigate performance in gaining the optimal solution.

Several members of the "Star Dome Truss" in example 4 of section 8.4 were sized to the design minimal area of .1 in^2. Table 16 lists the member results and shows members: 14, 15, 17, 18, 20, 21, 23, 24, 26, 27, 29 and 30 at the design minimal area. This is due to the light loads in the members or members that have no load in them and are passive. The same condition also exists for member 20 of the example in section 8.5. Sizing results for member 20 are listed in table 20 and show that the Khot area (.0524 in^2) is much less than the Hrinda obtained area (2.0824 in^2). The difference is from the Hrinda program enforcing a minimum area of .1 in^2 that caused the program to adjust other members to still achieve a final weight nearly equal to the Khot optimal weight. The condition may change as the applied loading is placed at different nodes on the structure or if the load direction is changed.

Modern design of lightweight truss structures has been achieved through the use of analysis techniques that were developed to investigate nonlinear instabilities. The application of loads to a truss system that produces nonproportional displacements is nonlinear; increases in the loading will make the structure unstable or cause it to buckle. In [21], these systems are called highly flexible structures (HFS) and are designed for large elastic displacements and without any plastic deformation. These HFS may either show bifurcation or limit-point instability. Bifurcation instability occurs when the structure has a sudden geometric change and is characterized by an abrupt equilibrium path change along the load path. Strain energy from a compressive force is converted to bending strain energy without a large force increase. This conversion can best be visualized as a typical Euler buckling of a slender column. The second type of nonlinear structural response is limit-point instability. This behavior is represented as a gradually

changing equilibrium path with no sudden changes. Increasing the load beyond the limit point causes a release of strain energy as the stiffness becomes negative.

Lightweight truss structures are often highly flexible with unpredictable nonlinear responses and may become unstable before a bifurcation point is reached. An understanding of elastic stability behavior in nonlinear structures can be gained by graphically representing the load and the displacement (i.e. the equilibrium path of the structure).

Methods that overcome the challenges of accurately identifying nonlinear equilibrium paths have been improving and have been developed to include optimization at limit points. With varying success, a growing number of attempts have been made to optimize a structure at a limit load. The challenge is to design a flexible structure to undergo large displacements and remain functional without a loss of stability.

CHAPTER X

CONCLUSIONS AND FUTURE RESEARCH

The optimization of stability-constrained truss structures that exhibit highly nonlinear snap-through behavior has been accomplished using the arc length method and the principal of strain energy density. The iterative scheme for updating design variables was used in a finite element approach (the proposed algorithm) that also used the critical buckling load at snap-through as the design load. The algorithm was able to find the optimum weight to prevent snap-through buckling and was used to size structures prone to snap-through. These included slender member systems such as reticulated shallow domes and large built-up arches. The problems that were executed agreed closely with previous truss examples that were found in the literature. The minimum weight was shown to exist when a uniform strain energy density was achieved in the members. The results verify that the use of the arc length parameters can be used as performance indicators to update the design variables. The technique was applied to design problems with highly nonlinear trusses that are prone to global instability failures, but the technique has also been tested on linear optimization problems. Initial results show that the arc length and strain energy density parameters can be used to find the lightest truss that exhibits linear behavior.

The exploration of space will require light weight structures to enable new technologies and missions. The highly flexible structures have minimal weight and a small stowed volume that make them desirable for space applications. Among NASA's

technology investment initiatives are large, on-orbit structures that support optical reflectors for solar power, communication networks and space telescopes. The structures must be assembled in orbit or designed to be easily deployed with a low risk of failure. Future methods for analyzing and measuring the nonlinear behavior of HFS will be required. Attempting to demonstrate these types of slender structures on Earth presents many challenges. Having tools that can accurately predict the structure's response in a space environment will improve conceptual designs and help direct research efforts.

A unique feature of the proposed Hrinda algorithm was the discovery of how it could be used to size MEMS in the electronic industry. Micro snap-through structures have been evolving as rapid switches that are integrated into solid-state circuits [2]. A difficult problem in their design is finding the limit point of the switch at snap-through. The author's algorithm should be able to size a micro-switch at snap-through but also for snap-back. Further investigation into the snap-back behavior of micro-structures may lead to new applications and MEMS devices.

The proposed Hrinda method is warranted because it is a self contained algorithm that automatically finds the lightest truss structure that supports a load just before a loss of stability. This capability of combining the arc length parameters to predict an update to the design variables is unique. The author has shown that his work takes an existing powerful nonlinear numerical technique, the arc length method, and uses its parameters to update a weight optimization problem.

The significant contribution of this work is to show how the arc-length method can be used to not only solve the nonlinear system but also to incorporate its parameters into an

optimization scheme. The author uses existing information from the arc-length method as it solves for the equilibrium path and applies this information to the weight minimization of the members. Dynamic inertial effects during the unstable region along the equilibrium path have been excluded by the author. According to the work of Alfutov [15] and Ziegler [28], a static approach leads to the same results as a dynamic approach.

The proposed Hrinda program also incorporates sparse computer techniques that allow for efficient use of computer resources [19]. This may lead to further development of the algorithm by not limiting the size of models and allowing more complicated shallow structures to be investigated.

Future work may involve the use of the Hrinda arc length program to investigate progressive failure in 3d space trusses. The ability of an inelastic analysis tool that confidently predicts limit points would be beneficial in designing orbital structures, transmission towers, bridges or structures that span long distances such as sports arenas [5]. By knowing how the internal load in these structures is redistributed due to damage or loss of stiffness in a member could help in the design of robust structures that are less prone to a catastrophic event. Modeling of post-buckled structures that can trace their equilibrium path from an initial limit point to a new stable point can give insight on system failures and what design improvements are necessary for surviving failure scenarios. If a structure can survive a dynamic snap-through event, it may be able to be designed for higher loads such as HFS presented by Pai [21].

BIBLIOGRAPHY

[1] Greenberg, H. S., "Large Space Structures," taken from lecture notes presented at NASA Langley Research Center, June 14–18, 2004.

[2] Blandford, G. E., "Progressive Failure Analysis of Inelastic Space Truss Structures," *Computers and Structures*, Vol. 58, 1996, pp. 981–990.

[3] Chollet, F. and Liu, H. B., "A (not so) Short Introduction to Micro Electromechanical Systems", version 3.0, 2009, http://memscyclopedia.org/introMEMS.html.

[4] Pane, I. Z. and Asano, T., "Investigation on Bistability and Fabrication of Bistable Prestressed Curved Beam," *Japanese Journal of Applied Physics*, Vol. 47, No. 6, 2008, pp. 5291–5296.

[5] Korvink, J. G. and Paul, O., *MEMS - A Practical Guide to Design, Analysis, and Applications*, William Andrew Publishing/Noyes, 2006.

[6] Prager, W., "Optimality Criteria in Structural Design," Proceedings of the National Academy of Sciences of the United States of America, Vol. 61, No. 3, 1968, pp. 794–796.

[7] Prager, W. and Taylor, J. E., "Problems of Optimal Structural Design," *J. Appl. Mech.*, Trans. ASME, Vol. 90, 1968, pp. 102–106.

[8] Khot, N. S., Venkayya, V. B., and Berke, L., "Optimum Structural Design With Stability Constraints," *Int. J. Num. Methods*, Vol. 10, 1976, pp. 1097–1114.

[9] Kamat, M. P. and Ruangsilasingha, P., "Design Sensitivity Derivatives for Structures in Nonlinear Response," AIAA-1984-0973.

[10] Crisfield, M. A., *Non-Linear Finite Element Analysis of Solids and Structures: Volume I— Essentials*, John Wiley & Sons, 1991.

[11] Crisfield, M. A., *Non-Linear Finite Element Analysis of Solids and Structures: Volume 2— Advanced Topics*, John Wiley & Sons, 1997.

[12] Warren, J. E., *Nonlinear Stability Analysis of Frame-Type Structures with Random Geometric Imperfections Using a Total-Lagrangian Finite Element Formulation,* Ph.D. Dissertation, Virginia Polytechnic Institute and State University, January 1997.

[13] Khot, N. S. and Kamat, M. P., "Minimum Weight Design of Truss Structures with Geometric Nonlinear Behavior," *AIAA Journal*, Vol. 23, No. 1, January 1985.

[14] Doyle, James F., *Nonlinear Analysis of Thin-Walled Structures*, Springer, 2001.

[15] Alfutov, N. A., *Stability of Elastic Structures*, Springer Publisher, 2000.

[16] Crisfield, M. A., "A Fast Incremental/Iterative Solution Procedure that Handles Snap-Through," *Comp. & Struct.*, Vol. 13, 1981, 55–62.

[17] Khot, N. S., Kamat, M. P. and Venkayya, V. B., "Optimization of Shallow Trusses Against Limit Point Instability," *AIAA Journal*, Vol. 22, No. 3, March 1984.

[18] Sedaghati, R., Tabarrok, B., and Suleman, A., "Optimum Design of Nonlinear Symmetric Truss Structure Under System Stability Constraint," 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA-2000-4835, Sept. 2000.

[19] Nguyen, D. T., *Finite Element Methods: Parallel-Sparse Statics and Eigen-Solutions*, Springer Publisher, 2006.

[20] Hrinda, G. A., "Geometrically Nonlinear Static Analysis of 3D Trusses Using the Arc-Length Method," *Computational Methods and Experimental Measurements XIII*, WIT Press, 2007, pp. 243-252.

[21] Pai, P. F., *Highly Flexible Structures: Modeling, Computation, and Experimentation*, AIAA Education Series, 2007.

[22] Xiao-zu, S. and Bashir-Ahmed, M., "Arc-Length Technique for Nonlinear Finite Element Analysis," *Journal of Zhejiang University Science*, 2004, pp. 618–628.

[23] West, H. and Geschwindner, L., *Fundamentals of Structural Analysis,* 2nd edition, John Wiley & Sons, 2002.

[24] Kollar, Lajos P., *Structural Stability in Engineering Practice*, E & FN Spon publishers, 1999.

[25] Geers, M. G. D., "Enhanced Solution Control for Physically and Geometrically Non-Linear Problems. Part II Comparative Performance Analysis," *Inter. Journal for Numerical Methods in Engineering*, Vol. 46, 1999, pp. 205–230.

[26] Suleman, A. and Sedaghati, R., "Benchmark case studies in optimization of geometrically nonlinear structures," *Structural and Multidisciplinary Optimization*, Vol. 30, 2005, pp. 273–296.

[27] Bruns, T. E., Sigmund, O., and Tortorelli, D. A., "Numerical Methods for the Topology Optimization of Structures that Exhibit Snap-Through," *Inter. Journal for Numerical Methods in Engineering*, Vol. 55, 2002, pp. 1215–1237.

[28] Ziegler, H., *Principles of Structural Stability*, Blaisdell, 1968.

[29] Thompson, J.M.T. and Hunt, G.W., *Elastic Instability Phenomena*, John Wiley & Sons Ltd., 1984.

APPENDIX A-Hrinda Rikarc Users' Input Data Description

A.1 Auro Function Example

$$10u_1^2 + 2u_2^2 = 0.0$$

$$2u_1^2 + 2u_2^2 = 0.0$$

$$\begin{pmatrix} 10u_1 & 2u_2 \\ 2u_1 & 2u_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}$$

The solution to the two unknowns (u_1, u_2) in (2.40) is (0,0).

The input file:

```
4   3   2   2   1   1   1   0  0  0  1  12  1
1  0  0  0  0  0
  6   12  24   4   4   4
1.e7   1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
 5.0  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1   0.        0.      0.
2   2500.    25.      0.
    1    1   2
1 1.d-8
2 1.d-8
  3  0.0
  4  0.0
  5  0.0
  6  0.0
```

The following are User changes made inside the program:

```
************************
c.....set "iex" to indicate that this is a function test
c.....This is done at beginning of the program
            iex=10 !Function Testing

c....User will edit "subroutine elstif" to suit the desired function
     subroutine elstif.....
            If(iex .eq. 10)then
c**Function Test**
C***Auroa's soln=(0,0)     **OK**
     esm(1,1) = 10.d0
     esm(1,2) = 2.d0
     esm(2,1) = 2.d0
     esm(2,2) = 2.d0
C***Arora's nonlinear soln=(0,0)  **OK**
    finx(1) = 10.d0*ux(1) + 2.d0*uy(1)
    finy(1) = 2.d0*ux(1) + 2.d0*uy(1)

c....User will edit "subroutine rikarc" to suit the function
c....being solved.  These are the arclength parameters that
c....produced the correct solutions.
            Subroutine rikarc.....
c***Arora's nonlinear page 331 soln=(0,0)
     dl0 = 1.0d-1
     dlmax = 1.0d-1
     dlmin = 1.0d-2
    exlmx = 1.d0
    itedes = 2
    toldis = 1.d-3
    tolfor = 1.d-4

c...User may edit "subroutine result" for desired output format
```

	istep	exlk	appliedforce1	appliedforce2	a(1)	a(2)	it	ad(5)	ad(7)
*	1	0.10000000E+00	0.1000E-08	0.1000E-08	0.0000E+00	0.5000E-09	2	0.1000E+01	0.1113-307
*	2	0.20000000E+00	0.2000E-08	0.2000E-08	0.0000E+00	0.1000E-08	3	0.1000E+01	0.1113-307
*	3	0.28164966E+00	0.2816E-08	0.2816E-08	0.0000E+00	0.1408E-08	3	0.1000E+01	0.1113-307
*	4	0.33938469E+00	0.3394E-08	0.3394E-08	0.0000E+00	0.1697E-08	3	0.1000E+01	0.1113-307
*	5	0.37589952E+00	0.3759E-08	0.3759E-08	0.0000E+00	0.1879E-08	3	0.1000E+01	0.1113-307
*	6	0.39698137E+00	0.3970E-08	0.3970E-08	0.0000E+00	0.1985E-08	3	0.1000E+01	0.1113-307
*	7	0.40825010E+00	0.4083E-08	0.4083E-08	0.0000E+00	0.2041E-08	3	0.1000E+01	0.1113-307
*	8	0.41825010E+00	0.4183E-08	0.4183E-08	0.0000E+00	0.2091E-08	3	0.1000E+01	0.1113-307
*	61	0.94825010E+00	0.9483E-08	0.9483E-08	0.2068E-26	0.4741E-08	2	0.1000E+01	0.1113-307
*	62	0.95825010E+00	0.9583E-08	0.9583E-08	0.2068E-26	0.4791E-08	2	0.1000E+01	0.1113-307
*	63	0.96825010E+00	0.9683E-08	0.9683E-08	0.2068E-26	0.4841E-08	2	0.1000E+01	0.1113-307
*	64	0.97825010E+00	0.9783E-08	0.9783E-08	0.0000E+00	0.4891E-08	2	0.1000E+01	0.1113-307
*	65	0.98825010E+00	0.9883E-08	0.9883E-08	0.0000E+00	0.4941E-08	2	0.1000E+01	0.1113-307
*	66	0.99825010E+00	0.9983E-08	0.9983E-08	0.0000E+00	0.4991E-08	2	0.1000E+01	0.1113-307
*	67	0.10082501E+01	0.1008E-07	0.1008E-07	0.0000E+00	0.5041E-08	2	0.1000E+01	0.1113-307

A.2 Auro 2 Function Example

$$24u_1 - 12u_2 = -2.0$$

$$-12u_1 + 8u_2 = 0.0$$

$$\begin{pmatrix} 24 & -12 \\ -12 & 8 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -2.0 \\ 0.0 \end{pmatrix}$$

The solution to the unknowns (u_1, u_2) is given as (-.3333,-.5000) respectively.

The input file:

```
4  3  2  2  1  1  1  0 0 0 1  12  1
1  0  0  0  0  0
 6  12 24  4  4  4
1.e7  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
 5.0  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1  0.      0.     0.
2  2500.   25.    0.
   1   1  2
1 -2.d0
2 0.d0
 3  0.0
 4  0.0
 5  0.0
 6  0.0
```

The following are User changes made inside the program:

```
**********
c.....set "iex" to indicate that this is a function test
c.....This is done at begining of the program
            iex=10 !Function Testing

c....User will edit "subroutine elstif" to suit the desired function
      subroutine elstif.....
            If(iex .eq. 10)then
c**Function Test**
C***Arora's nonlinear 2  soln=(-.33, -.5)    **OK**
      esm(1,1) = 24.d0
      esm(1,2) = -12.d0
      esm(2,1) = -12.d0
      esm(2,2) = 8.d0
      Endif
```

```
**********
C***Arora's nonlinear 2 sol=(-.33, -.5)  **OK**
     finx(1) = 24.d0*ux(1) - 12.d0*uy(1)
     finy(1) = -12.d0*ux(1) + 8.d0*uy(1)

c....User will edit "subroutine rikarc" to suit the function
c....being solved.  These are the arclength parameters that
c....produced the correct solutions.
              Subroutine rikarc.....
c>>> Arora's nonlinear 2 ex={-1/3,-1/2} <<<              OK
       dl0 = 1.0d-1
       dlmax = 1.0d-1
       dlmin = 1.0d-2
      exlmx = 1.d0
      itedes = 2
      toldis = 1.d-3
      tolfor = 1.d-4

c...User may edit "subroutine result" for desired output format
```

	istep	exlk	appliedforce1	appliedforce	a(1)	a(2)	it	ad(5)	ad(7)
*	1	0.10000000E+00	-.2000E+00	0.0000E+00	-.3333E-01	-.5000E-01	2	0.1000E+01	0.1113-307
*	2	0.21280366E+00	-.4256E+00	0.0000E+00	-.7093E-01	-.1064E+00	3	0.1000E+01	0.1113-307
*	3	0.30198417E+00	-.6040E+00	0.0000E+00	-.1007E+00	-.1510E+00	3	0.1000E+01	0.1113-307
*	4	0.36371055E+00	-.7274E+00	0.0000E+00	-.1212E+00	-.1819E+00	3	0.1000E+01	0.1113-307
*	5	0.40225448E+00	-.8045E+00	0.0000E+00	-.1341E+00	-.2011E+00	3	0.1000E+01	0.1113-307
*	6	0.42434893E+00	-.8487E+00	0.0000E+00	-.1414E+00	-.2122E+00	3	0.1000E+01	0.1113-307
*	7	0.43611369E+00	-.8722E+00	0.0000E+00	-.1454E+00	-.2181E+00	3	0.1000E+01	0.1113-307
*	8	0.44653339E+00	-.8931E+00	0.0000E+00	-.1488E+00	-.2233E+00	3	0.1000E+01	0.1113-307
*	9	0.45693546E+00	-.9139E+00	0.0000E+00	-.1523E+00	-.2285E+00	3	0.1000E+01	0.1113-307
*	10	0.46732039E+00	-.9346E+00	0.0000E+00	-.1558E+00	-.2337E+00	3	0.1000E+01	0.1113-307
*	11	0.47768867E+00	-.9554E+00	0.0000E+00	-.1592E+00	-.2388E+00	3	0.1000E+01	0.1113-307
*	12	0.48804077E+00	-.9761E+00	0.0000E+00	-.1627E+00	-.2440E+00	3	0.1000E+01	0.1113-307
*	13	0.49837716E+00	-.9968E+00	0.0000E+00	-.1661E+00	-.2492E+00	3	0.1000E+01	0.1113-307
*	14	0.50869831E+00	-.1017E+01	0.0000E+00	-.1696E+00	-.2543E+00	3	0.1000E+01	0.1113-307
*	15	0.51900466E+00	-.1038E+01	0.0000E+00	-.1730E+00	-.2595E+00	3	0.1000E+01	0.1113-307
*	16	0.52929667E+00	-.1059E+01	0.0000E+00	-.1764E+00	-.2646E+00	3	0.1000E+01	0.1113-307
*	17	0.53957476E+00	-.1079E+01	0.0000E+00	-.1799E+00	-.2698E+00	3	0.1000E+01	0.1113-307
*	18	0.54983936E+00	-.1100E+01	0.0000E+00	-.1833E+00	-.2749E+00	3	0.1000E+01	0.1113-307
*	19	0.56009090E+00	-.1120E+01	0.0000E+00	-.1867E+00	-.2800E+00	3	0.1000E+01	0.1113-307
*	20	0.57032978E+00	-.1141E+01	0.0000E+00	-.1901E+00	-.2852E+00	3	0.1000E+01	0.1113-307
*	21	0.58055642E+00	-.1161E+01	0.0000E+00	-.1935E+00	-.2903E+00	3	0.1000E+01	0.1113-307
*	22	0.59077120E+00	-.1182E+01	0.0000E+00	-.1969E+00	-.2954E+00	3	0.1000E+01	0.1113-307
*	23	0.60097453E+00	-.1202E+01	0.0000E+00	-.2003E+00	-.3005E+00	3	0.1000E+01	0.1113-307
*	24	0.61116678E+00	-.1222E+01	0.0000E+00	-.2037E+00	-.3056E+00	3	0.1000E+01	0.1113-307
*	25	0.62134833E+00	-.1243E+01	0.0000E+00	-.2071E+00	-.3107E+00	3	0.1000E+01	0.1113-307
*	26	0.63151955E+00	-.1263E+01	0.0000E+00	-.2105E+00	-.3158E+00	3	0.1000E+01	0.1113-307
*	27	0.64168081E+00	-.1283E+01	0.0000E+00	-.2139E+00	-.3208E+00	3	0.1000E+01	0.1113-307
*	28	0.65183246E+00	-.1304E+01	0.0000E+00	-.2173E+00	-.3259E+00	3	0.1000E+01	0.1113-307
*	29	0.66197485E+00	-.1324E+01	0.0000E+00	-.2207E+00	-.3310E+00	3	0.1000E+01	0.1113-307
*	30	0.67210834E+00	-.1344E+01	0.0000E+00	-.2240E+00	-.3361E+00	3	0.1000E+01	0.1113-307
*	31	0.68223326E+00	-.1364E+01	0.0000E+00	-.2274E+00	-.3411E+00	3	0.1000E+01	0.1113-307
*	32	0.69234995E+00	-.1385E+01	0.0000E+00	-.2308E+00	-.3462E+00	3	0.1000E+01	0.1113-307
*	33	0.70245873E+00	-.1405E+01	0.0000E+00	-.2342E+00	-.3512E+00	3	0.1000E+01	0.1113-307
*	34	0.71255992E+00	-.1425E+01	0.0000E+00	-.2375E+00	-.3563E+00	3	0.1000E+01	0.1113-307
*	35	0.72265384E+00	-.1445E+01	0.0000E+00	-.2409E+00	-.3613E+00	3	0.1000E+01	0.1113-307
*	36	0.73274081E+00	-.1465E+01	0.0000E+00	-.2442E+00	-.3664E+00	2	0.1000E+01	0.1113-307
*	37	0.74282113E+00	-.1486E+01	0.0000E+00	-.2476E+00	-.3714E+00	2	0.1000E+01	0.1113-307
*	38	0.75289511E+00	-.1506E+01	0.0000E+00	-.2510E+00	-.3764E+00	2	0.1000E+01	0.1113-307
*	39	0.76296303E+00	-.1526E+01	0.0000E+00	-.2543E+00	-.3815E+00	2	0.1000E+01	0.1113-307
*	40	0.77302519E+00	-.1546E+01	0.0000E+00	-.2577E+00	-.3865E+00	2	0.1000E+01	0.1113-307
*	41	0.78308188E+00	-.1566E+01	0.0000E+00	-.2610E+00	-.3915E+00	2	0.1000E+01	0.1113-307
*	42	0.79313338E+00	-.1586E+01	0.0000E+00	-.2644E+00	-.3966E+00	2	0.1000E+01	0.1113-307
*	43	0.80317996E+00	-.1606E+01	0.0000E+00	-.2677E+00	-.4016E+00	2	0.1000E+01	0.1113-307
*	44	0.81322190E+00	-.1626E+01	0.0000E+00	-.2711E+00	-.4066E+00	2	0.1000E+01	0.1113-307
*	45	0.82325947E+00	-.1647E+01	0.0000E+00	-.2744E+00	-.4116E+00	2	0.1000E+01	0.1113-307
*	46	0.83329292E+00	-.1667E+01	0.0000E+00	-.2778E+00	-.4166E+00	2	0.1000E+01	0.1113-307
*	47	0.84332253E+00	-.1687E+01	0.0000E+00	-.2811E+00	-.4217E+00	2	0.1000E+01	0.1113-307
*	48	0.85334854E+00	-.1707E+01	0.0000E+00	-.2844E+00	-.4267E+00	2	0.1000E+01	0.1113-307
*	49	0.86337121E+00	-.1727E+01	0.0000E+00	-.2878E+00	-.4317E+00	2	0.1000E+01	0.1113-307

*	50	0.87339077E+00	-.1747E+01	0.0000E+00	-.2911E+00	-.4367E+00	2	0.1000E+01	0.1113-307
*	51	0.88340749E+00	-.1767E+01	0.0000E+00	-.2945E+00	-.4417E+00	2	0.1000E+01	0.1113-307
*	52	0.89342159E+00	-.1787E+01	0.0000E+00	-.2978E+00	-.4467E+00	2	0.1000E+01	0.1113-307
*	53	0.90343330E+00	-.1807E+01	0.0000E+00	-.3011E+00	-.4517E+00	2	0.1000E+01	0.1113-307
*	54	0.91344287E+00	-.1827E+01	0.0000E+00	-.3045E+00	-.4567E+00	2	0.1000E+01	0.1113-307
*	55	0.92345052E+00	-.1847E+01	0.0000E+00	-.3078E+00	-.4617E+00	2	0.1000E+01	0.1113-307
*	56	0.93345647E+00	-.1867E+01	0.0000E+00	-.3112E+00	-.4667E+00	2	0.1000E+01	0.1113-307
*	57	0.94346094E+00	-.1887E+01	0.0000E+00	-.3145E+00	-.4717E+00	2	0.1000E+01	0.1113-307
*	58	0.95346415E+00	-.1907E+01	0.0000E+00	-.3178E+00	-.4767E+00	2	0.1000E+01	0.1113-307
*	59	0.96346631E+00	-.1927E+01	0.0000E+00	-.3212E+00	-.4817E+00	2	0.1000E+01	0.1113-307
*	60	0.97346764E+00	-.1947E+01	0.0000E+00	-.3245E+00	-.4867E+00	2	0.1000E+01	0.1113-307
*	61	0.98346833E+00	-.1967E+01	0.0000E+00	-.3278E+00	-.4917E+00	2	0.1000E+01	0.1113-307
*	62	0.99346860E+00	-.1987E+01	0.0000E+00	-.3312E+00	-.4967E+00	2	0.1000E+01	0.1113-307
*	63	0.10034686E+01	-.2007E+01	0.0000E+00	-.3345E+00	-.5017E+00	2	0.1000E+01	0.1113-307

A.3 Hrinda 1 Function Example

$$24u_1^2 - 12u_2 = 0$$

$$-12u_1 + 8u_2^2 = 20$$

$$\begin{pmatrix} 24u_1 & -12 \\ -12 & 8u_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 20 \end{pmatrix}$$

The unknowns, (u_1, u_2) in (2.28) have a unique solution that is (-.7633, 1.165).

The input file:

```
4   3   2   2   1   1   1   0  0  0  1  12  1
1  0  0  0  0  0
  6   12  24   4   4   4
1.e7   1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
 5.0  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1   0.        0.       0.
2   2500.     25.      0.
    1    1   2
1 0.d0
2 20.d0
  3  0.0
  4  0.0
  5  0.0
  6  0.0
```

The following are User changes made inside the program:

```
************************
c.....set "iex" to indicate that this is a function test
c.....This is done at begining of the program
          iex=10 !Function Testing
c....User will edit "subroutine elstif" to suit the desired function
c***Hrinda 1 (0.,20.)  OK  **OK**(-.7633, 1.165)
      subroutine elstif.....
          If(iex .eq. 10)then
c**Function Test**
      esm(1,1) = 24.d0*ux(1) + 2.d-6
      esm(1,2) = -12.d0 + 1.d-6
      esm(2,1) = -12.d0 + 1.d-6
      esm(2,2) = 8.d0*uy(1) + 2.d-6
      Endif
c***Hrinda 1   **OK**
       finx(1)=(24.d0*ux(1)+2.d-6)*ux(1)+(-12.d0+1.d-6)*uy(1)
       finy(1)=(-12.d0+1.d-6)*ux(1)+(8.d0*uy(1)+2.d-6)*uy(1)
```

```
c....User will edit "subroutine rikarc" to suit the function
c....being solved.  These are the arclength parameters that
c....produced the correct solutions.
          Subroutine rikarc.....
c***Hrinda 1              **OK**
      dl0 = 1.d-2
      dlmax = 1.d-3
      dlmin = 1.d-4
     exlmx = 1.d0
     itedes = 2
     toldis = 1.d-3
     tolfor = 1.d-4

c...User may edit "subroutine result" for desired output format
```

	istep	exlk	appliedforce1	appliedforce2	a(1)	a(2)	it	ad(5)	ad(7)
*	1	0.10000124E-01	0.0000E+00	0.2000E+00	-.1667E-01	0.5556E-03	3	0.1000E+01	0.1113-307
*	2	0.11891384E-01	0.0000E+00	0.2378E+00	-.1982E-01	0.7856E-03	3	0.1000E+01	0.1113-307
*	3	0.13427767E-01	0.0000E+00	0.2686E+00	-.2238E-01	0.1002E-02	3	0.1000E+01	0.1113-307
*	4	0.14509699E-01	0.0000E+00	0.2902E+00	-.2418E-01	0.1170E-02	3	0.1000E+01	0.1113-307
*	5	0.15192002E-01	0.0000E+00	0.3038E+00	-.2532E-01	0.1282E-02	3	0.1000E+01	0.1113-307
....									
....									
*	7613	0.99924043E+00	0.0000E+00	0.1998E+02	-.7628E+00	0.1164E+01	1	0.1000E+01	0.1113-307
*	7614	0.99942254E+00	0.0000E+00	0.1999E+02	-.7628E+00	0.1164E+01	2	0.1000E+01	0.1113-307
*	7615	0.99953597E+00	0.0000E+00	0.1999E+02	-.7629E+00	0.1164E+01	1	0.1000E+01	0.1113-307
*	7616	0.99971813E+00	0.0000E+00	0.1999E+02	-.7629E+00	0.1164E+01	2	0.1000E+01	0.1113-307
*	7617	0.99983154E+00	0.0000E+00	0.2000E+02	-.7629E+00	0.1164E+01	1	0.1000E+01	0.1113-307
*	7618	0.10000138E+01	0.0000E+00	0.2000E+02	-.7629E+00	0.1164E+01	2	0.1000E+01	0.1113-307

A.4 Hrinda 3 Function Example

$$24u_1^3 - 16u_2 = 8$$

$$-36u_1 + 6u_2^2 = -24$$

$$\begin{pmatrix} 24u_1^2 & -16 \\ -36 & 6u_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 8 \\ -24 \end{pmatrix}$$

The solution to the two unknowns is (.6675, -.05417).

The input file:

```
4   3   2   2   1   1   1   0  0  0  1  12  1
1  0  0  0  0  0
  6   12  24   4   4   4
1.e7   1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
 5.0  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1   0.        0.       0.
2   2500.     25.      0.
    1    1   2
1 8.d0
2 -24.d0
  3  0.0
  4  0.0
  5  0.0
  6  0.0
```

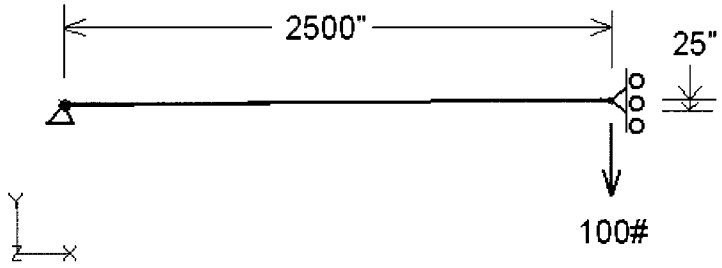
The following are User changes made inside the program:
****

```
c.....set "iex" to indicate that this is a function test
c.....This is done at beginning of the program
```

```
            iex=10 !Function Testing

c....User will edit "subroutine elstif" to suit the desired function
c***Hrinda 3 (6675, -.05417.)  **OK**
      esm(1,1) =  24.d0*ux(1)*ux(1) + 2.d-10
      esm(1,2) =  -16.d0
      esm(2,1) =  -36.d0
      esm(2,2) =  6.d0*uy(1) + 2.d-10

c***Hrinda 3   **OK**
       finx(1)=(24.d0*ux(1)*ux(1)+2.d-10)*ux(1)+
     c        (-16.d0)*uy(1)

       finy(1)=(-36.d0)*ux(1)+
     c        (6.d0*uy(1)+2.d-10)*uy(1)

c....User will edit "subroutine rikarc" to suit the function
c....being solved.  These are the arclength parameters that
c....produced the correct solutions.
              Subroutine rikarc.....
c***Hrinda 3              **OK**
       dl0 = 1.d-1
       dlmax = 1.d-2
       dlmin = 1.d-3
      exlmx = 1.d0
      itedes = 2
      toldis = 1.d-4
      tolfor = 1.d-5

c...User may edit "subroutine result" for desired output format
```

	istep	exlk	appliedforce1	appliedforce2	a(1)	a(2)	it	ad(5)	ad(7)
*	1	0.19961451E+00	0.1597E+01	-.4791E+01	0.1346E+00	-.9615E-01	7	0.1000E+01	0.1113-307
*	2	0.21535532E+00	0.1723E+01	-.5169E+01	0.1453E+00	-.1031E+00	60	0.1000E+01	0.1113-307
*	3	0.22811530E+00	0.1825E+01	-.5475E+01	0.1540E+00	-.1086E+00	50	0.1000E+01	0.1113-307
*	4	0.23709132E+00	0.1897E+01	-.5690E+01	0.1602E+00	-.1124E+00	43	0.1000E+01	0.1113-307
*	5	0.24274855E+00	0.1942E+01	-.5826E+01	0.1640E+00	-.1148E+00	37	0.1000E+01	0.1113-307
*	6	0.24600811E+00	0.1968E+01	-.5904E+01	0.1663E+00	-.1161E+00	33	0.1000E+01	0.1113-307
*	7	0.24774850E+00	0.1982E+01	-.5946E+01	0.1674E+00	-.1168E+00	29	0.1000E+01	0.1113-307
*	8	0.24929112E+00	0.1994E+01	-.5983E+01	0.1685E+00	-.1175E+00	28	0.1000E+01	0.1113-307
*	483	0.99015099E+00	0.7921E+01	-.2376E+02	0.6608E+00	-.6236E-01	9	0.1000E+01	0.1113-307
*	484	0.99163279E+00	0.7933E+01	-.2380E+02	0.6617E+00	-.6121E-01	9	0.1000E+01	0.1113-307
*	485	0.99311261E+00	0.7945E+01	-.2383E+02	0.6627E+00	-.6005E-01	9	0.1000E+01	0.1113-307
*	486	0.99459605E+00	0.7957E+01	-.2387E+02	0.6636E+00	-.5887E-01	10	0.1000E+01	0.1113-307
*	487	0.99607197E+00	0.7969E+01	-.2391E+02	0.6646E+00	-.5770E-01	10	0.1000E+01	0.1113-307
*	488	0.99754589E+00	0.7980E+01	-.2394E+02	0.6656E+00	-.5653E-01	10	0.1000E+01	0.1113-307
*	489	0.99901778E+00	0.7992E+01	-.2398E+02	0.6665E+00	-.5535E-01	10	0.1000E+01	0.1113-307
*	490	0.10004876E+01	0.8004E+01	-.2401E+02	0.6675E+00	-.5417E-01	10	0.1000E+01	0.1113-307

A.5 SDOF Crisfield Example



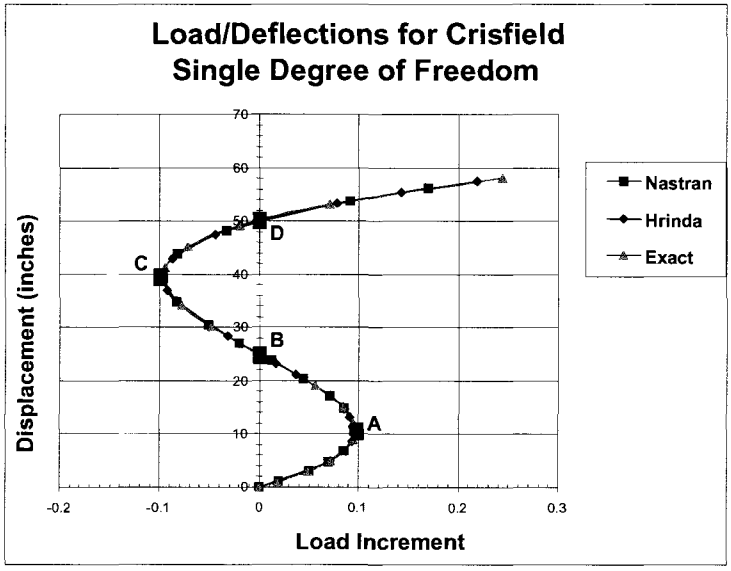

```
*************************
Input file:

5   3   2   1    1   1   1   0  0  0  1  12  1
1  0  0  0  0  0
  6   12  24   4   4   4
1.e7   1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
 5.0  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1   0.        0.       0.
2   2500.     25.      0.
    1    1   2
5 -100.d0
  1  0.0
  2  0.0
  3  0.0
  4  0.0
  6  0.0
*************************
```

The following are User changes made inside the program:
c....User will edit "subroutine rikarc" to suit the function
c....being solved. These are the arclength parameters that
c....produced the correct solutions.

Subroutine rikarc.....

```
c** Crisfield sdof Bar Truss
C** Symmetric
C** This is 2 nodes, 1 element
       itedes = 1
        dl0 = .001d0
        dlmax = .001d0
         dlmin = .0001d0
       exlmx = 1.d0
      toldis = 1.d-4
      tolfor = 1.d-4
```

c...User may edit "subroutine result" for desired output format



METHOD	A	B	C	D
NASTRAN	.0935, 12.59	0.000, 25.00	-.0962, 39.26	.0077, 50.38
Hrinda	.0948, 12.00	0.000, 24.94	-.0961, 39.00	.0070, 50.34
Exact	.0948, 12.00	0.000, 24.82	-.0962, 39.26	.0077, 50.38

	istep	exlk	appliedforce	a(5)	it	ad(4)	ad(5)	ad(6)
*	1	0.997002E-03	-.9970E-01	-.5001E-01	2	0.1000E+01	0.1988E+01	0.1000E+01
*	2	0.108259E-01	-.1083E+01	-.5601E+00	4	0.1000E+01	0.1867E+01	0.1000E+01
*	3	0.115603E-01	-.1156E+01	-.5995E+00	3	0.1000E+01	0.1858E+01	0.1000E+01
*	2203	0.948627E-01	-.9486E+01	-.1198E+02	2	0.1000E+01	-.1868E+00	0.1000E+01
*	2204	0.948534E-01	-.9485E+01	-.1199E+02	2	0.1000E+01	-.1875E+00	0.1000E+01
*	2205	0.948440E-01	-.9484E+01	-.1199E+02	2	0.1000E+01	-.1881E+00	0.1000E+01
*	2206	0.948345E-01	-.9483E+01	-.1200E+02	2	0.1000E+01	-.1887E+00	0.1000E+01
*	2207	0.948251E-01	-.9483E+01	-.1200E+02	2	0.1000E+01	-.1893E+00	0.1000E+01
*	2208	0.948156E-01	-.9482E+01	-.1201E+02	2	0.1000E+01	-.1900E+00	0.1000E+01
*	4623	0.167073E-02	-.1671E+00	-.2483E+02	3	0.1000E+01	-.9997E+00	0.1000E+01
*	4624	0.138450E-02	-.1384E+00	-.2486E+02	3	0.1000E+01	-.9998E+00	0.1000E+01
*	4625	0.104450E-02	-.1044E+00	-.2490E+02	3	0.1000E+01	-.9998E+00	0.1000E+01
*	4626	0.605247E-03	-.6052E-01	-.2494E+02	3	0.1000E+01	-.9998E+00	0.1000E+01
*	4627	-0.105074E-03	0.1051E-01	-.2501E+02	3	0.1000E+01	-.9998E+00	0.1000E+01
*	4628	-0.502351E-01	0.5024E+01	-.3026E+02	7	0.1000E+01	-.8672E+00	0.1000E+01
*	4629	-0.502791E-01	0.5028E+01	-.3026E+02	2	0.1000E+01	-.8670E+00	0.1000E+01
*	4630	-0.503232E-01	0.5032E+01	-.3027E+02	2	0.1000E+01	-.8667E+00	0.1000E+01
*	6364	-0.960673E-01	0.9607E+01	-.3898E+02	2	0.1000E+01	-.6234E-01	0.1000E+01
*	6365	-0.960704E-01	0.9607E+01	-.3898E+02	2	0.1000E+01	-.6167E-01	0.1000E+01
*	6366	-0.960735E-01	0.9607E+01	-.3899E+02	2	0.1000E+01	-.6100E-01	0.1000E+01
*	6367	-0.960765E-01	0.9608E+01	-.3899E+02	2	0.1000E+01	-.6033E-01	0.1000E+01
*	6368	-0.960795E-01	0.9608E+01	-.3900E+02	2	0.1000E+01	-.5965E-01	0.1000E+01
*	6369	-0.960825E-01	0.9608E+01	-.3900E+02	2	0.1000E+01	-.5898E-01	0.1000E+01
*	6370	-0.960854E-01	0.9609E+01	-.3901E+02	2	0.1000E+01	-.5831E-01	0.1000E+01
*	6371	-0.960883E-01	0.9609E+01	-.3901E+02	2	0.1000E+01	-.5764E-01	0.1000E+01
*	6372	-0.960912E-01	0.9609E+01	-.3902E+02	2	0.1000E+01	-.5696E-01	0.1000E+01
*	6373	-0.960940E-01	0.9609E+01	-.3902E+02	2	0.1000E+01	-.5629E-01	0.1000E+01
*	8406	0.653863E-02	-.6539E+00	-.5032E+02	3	0.1000E+01	0.2077E+01	0.1000E+01
*	8407	0.686351E-02	-.6864E+00	-.5034E+02	3	0.1000E+01	0.2081E+01	0.1000E+01

* 8408	0.717635E-02	-.7176E+00	-.5035E+02	3	0.1000E+01	0.2085E+01	0.1000E+01
* 8409	0.747862E-02	-.7479E+00	-.5037E+02	3	0.1000E+01	0.2088E+01	0.1000E+01
* 8410	0.777155E-02	-.7772E+00	-.5038E+02	3	0.1000E+01	0.2092E+01	0.1000E+01
* 8411	0.805611E-02	-.8056E+00	-.5039E+02	3	0.1000E+01	0.2095E+01	0.1000E+01
* 8412	0.833315E-02	-.8333E+00	-.5041E+02	3	0.1000E+01	0.2098E+01	0.1000E+01
* 8413	0.860338E-02	-.8603E+00	-.5042E+02	3	0.1000E+01	0.2101E+01	0.1000E+01

The sdof problem is gradually loaded in increments until point "A" is reached. Figure I illustrates this relationship and shows a stable structure up to point "A". A slight load change just after this point will cause the structure to lose stability and snap to a new equilibrium point "E." Point "A" is the stability "limit point" of the structure and is indicated by a loss of tangent stiffness and the slope of the load vs. displacement decreasing to zero. Internal loads and strains increase until point "A" is reached. During "snap-through" from point "A" to "E" the structure releases the stored strain energy. At point "E" the load continues to be applied as the structure deforms along a stable path.

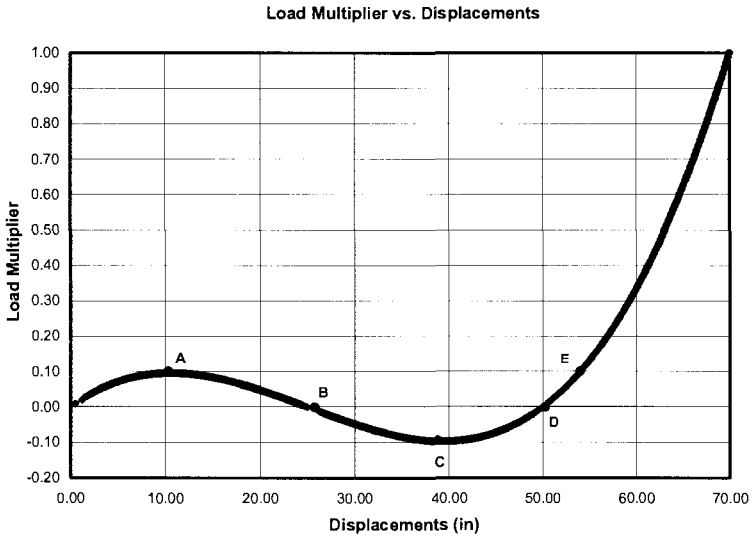


Figure I. Sdof load vs. displacement equilibrium path.

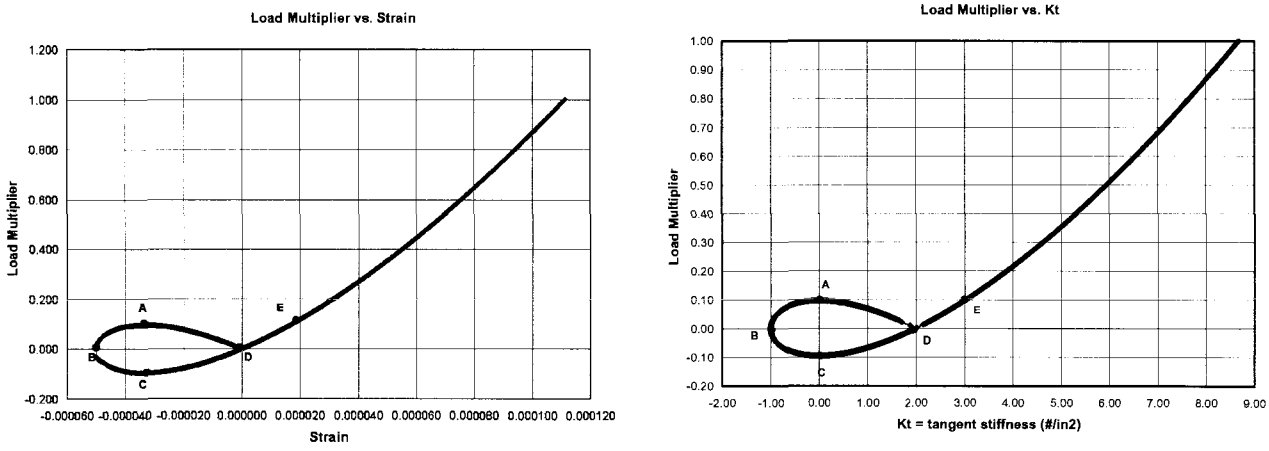
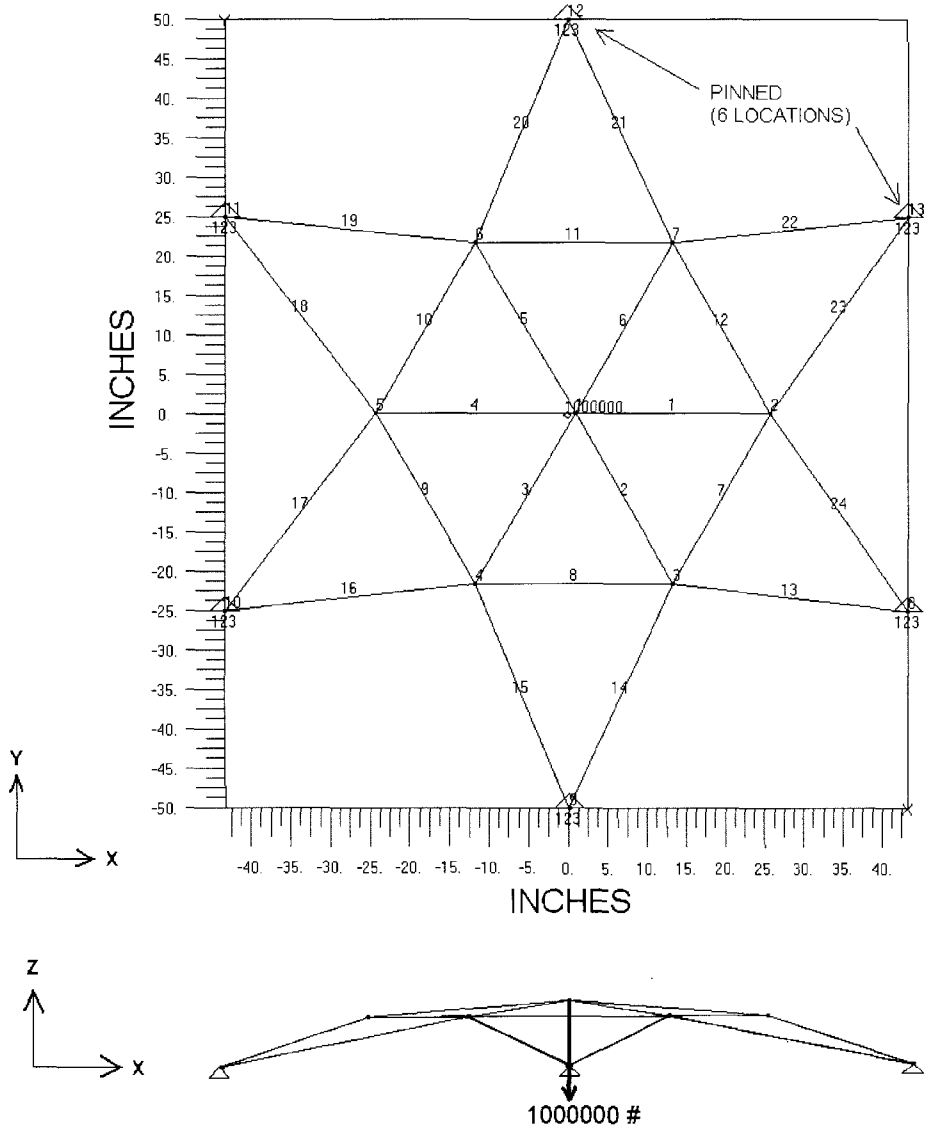


Figure II. Tangent stiffness and Strain as a function of the load multiplier.

A.6 Star Dome Crisfield Example



$E = 1.e7\ \#/in^2$

$A = 5.in^2$

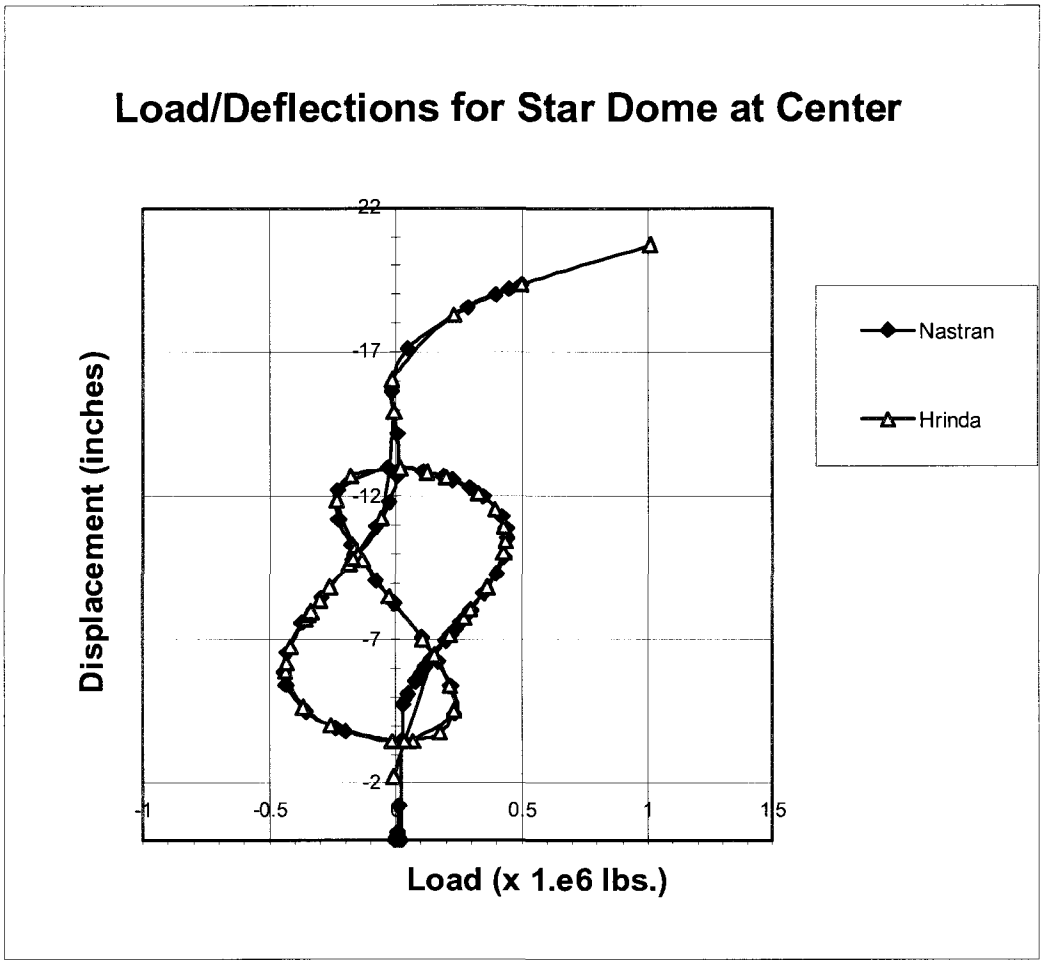
Crisfield Three-dimensional star dome.

```
************************
Input file:
18 3 13   1   1   1   1   0  0  0  1  12  1
24 0  0  0  0  0
  6   12  24   4   4   4
1.e7   1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
5.0  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1   0.      0.0     8.216
2   25.     0.0     6.216
3   12.5  -21.6506  6.216
4  -12.5  -21.6506  6.216
5   -25.    0.0     6.216
6   -12.5  21.6506  6.216
7   12.5   21.6506  6.216
```

```
8   43.3013 -25.     0.0
9   0.0     -50.     0.0
10 -43.3013 -25.     0.0
11 -43.3013  25.     0.0
12  0.0      50.     0.0
13  43.3013  25.     0.0
    1    1   2
    2    1   3
    3    4   1
    4    5   1
    5    6   1
    6    1   7
    7    3   2
    8    4   3
    9    5   4
   10    5   6
   11    6   7
   12    7   2
   13    3   8
   14    9   3
   15    4   9
   16   10   4
   17   10   5
   18   11   5
   19   11   6
   20    6  12
   21   12   7
   22    7  13
   23    2  13
   24    2   8
 3   -1.d6
  22    0.0
  23    0.0
  24    0.0
  25    0.0
  26    0.0
  27    0.0
  28    0.0
  29    0.0
  30    0.0
  31    0.0
  32    0.0
  33    0.0
  34    0.0
  35    0.0
  36    0.0
  37    0.0
  38    0.0
  39    0.0
************************
```

The following are User changes made inside the program:

```
c....User will edit "subroutine rikarc" to suit the function
c....being solved.  These are the arclength parameters that
c....produced the correct solutions.
            Subroutine rikarc.....
C** 3D Truss , Dome_in.dat & Dome2_in.dat
C** This is a 13 nodes, 24 elements,
      itedes=1
       dl0 = .1d0
       dlmax = .1d0
        dlmin = .001d0
      exlmx = 1.d0
     toldis = 1.d-6
     tolfor = 1.d-6
```

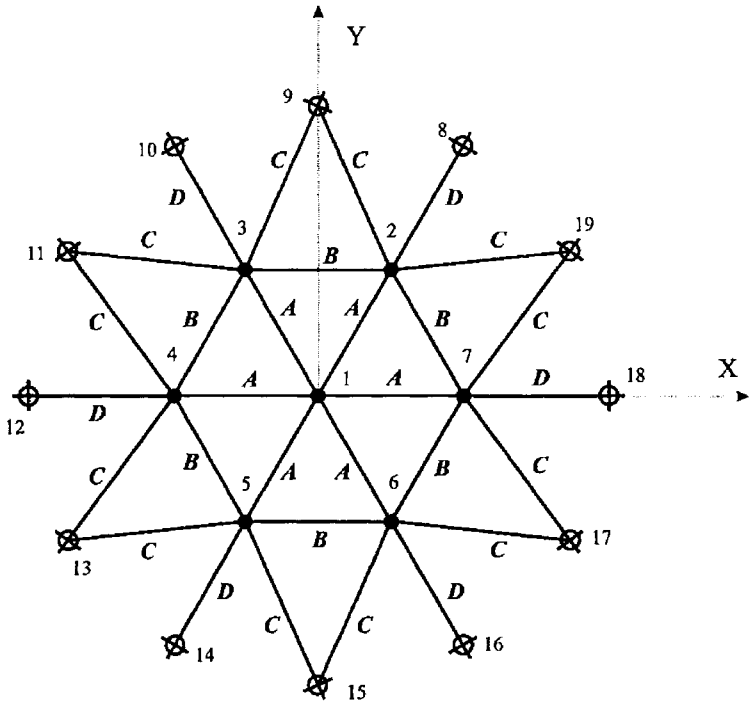


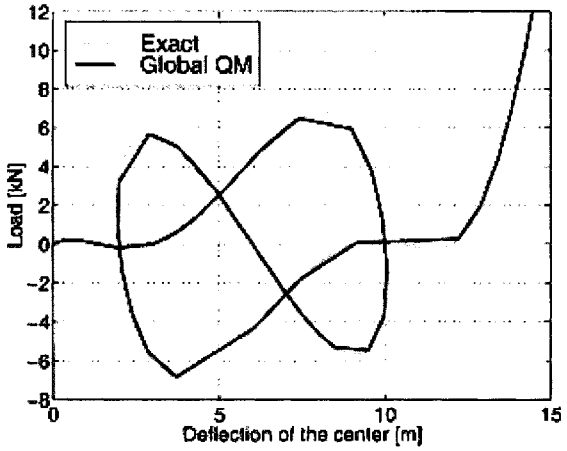
NASTRAN/Hrinda star dome result at center load increment vs. vertical displacement.

	istep	exlk	appliedforce	a(1)	a(2)	a(3)	a(4)	a(5)	a(6)	it
*	1	-0.613198E-02	0.61E+04	0.8176E-17	0.9569E-17	-.2214E+01	0.3939E-01	-.8266E-19	0.1228E+00	3
*	2	0.153542E+00	-0.15E+06	-.4192E-16	-.5105E-17	-.6455E+01	-.1267E+00	-.4306E-17	-.6402E+00	6
*	3	0.215746E+00	-0.22E+06	0.1489E-15	0.2964E-17	-.7139E+01	-.1701E+00	0.6667E-17	-.9146E+00	3
*	239	0.435728E+00	-0.44E+06	0.1643E-15	-.1986E-15	-.1048E+02	-.3753E+00	-.6752E-17	-.2923E+01	2
*	240	0.435747E+00	-0.44E+06	-.1700E-15	0.7847E-16	-.1049E+02	-.3758E+00	0.2872E-16	-.2931E+01	2
*	241	0.435759E+00	-0.44E+06	-.1185E-15	-.5368E-16	-.1050E+02	-.3763E+00	-.3975E-16	-.2940E+01	2
*	242	0.435764E+00	-0.44E+06	0.2131E-15	-.3607E-16	-.1051E+02	-.3768E+00	0.4574E-16	-.2948E+01	2
*	243	0.435763E+00	-0.44E+06	-.1309E-15	0.5790E-16	-.1052E+02	-.3773E+00	0.2716E-16	-.2956E+01	2
*	244	0.435755E+00	-0.44E+06	-.1580E-15	-.1310E-15	-.1053E+02	-.3777E+00	0.5843E-17	-.2964E+01	2
*	413	0.340246E+00	-0.34E+06	0.6311E-16	0.3436E-16	-.1199E+02	-.4250E+00	0.2277E-16	-.4385E+01	2
*	414	0.339142E+00	-0.34E+06	0.1600E-15	0.5224E-16	-.1200E+02	-.4251E+00	0.1863E-16	-.4394E+01	2
*	524	0.180440E+00	-0.18E+06	0.1668E-15	0.2785E-16	-.1268E+02	-.4148E+00	-.9725E-16	-.5375E+01	3
*	525	0.178632E+00	-0.18E+06	0.2451E-16	-.4119E-16	-.1269E+02	-.4145E+00	0.8214E-18	-.5385E+01	3
*	589	0.155746E-01	-0.16E+05	-.1664E-16	-.2999E-16	-.1296E+02	-.3794E+00	-.1122E-17	-.6149E+01	3
*	590	0.730061E-02	-0.73E+04	0.1501E-17	-.3359E-16	-.1297E+02	-.3771E+00	0.7526E-16	-.6185E+01	4
*	591	-0.515021E-02	0.52E+04	0.1629E-17	-.7377E-16	-.1297E+02	-.3735E+00	-.9992E-16	-.6238E+01	4
*	592	-0.541731E-01	0.54E+05	-.3265E-17	0.7351E-17	-.1297E+02	-.3580E+00	0.2697E-16	-.6441E+01	4
*	593	-0.599799E-01	0.60E+05	0.1265E-15	-.8314E-16	-.1297E+02	-.3560E+00	-.2009E-16	-.6464E+01	3
*	669	-0.232823E+00	0.23E+06	0.3174E-17	-.3527E-16	-.1182E+02	-.2507E+00	-.9562E-17	-.7070E+01	2
*	670	-0.232867E+00	0.23E+06	0.4125E-16	0.2075E-16	-.1180E+02	-.2498E+00	0.2450E-16	-.7069E+01	2

* 671	-0.232871E+00	0.23E+06	0.3503E-16	0.9808E-16	-.1178E+02	-.2489E+00	0.1705E-16	-.7068E+01	2
* 672	-0.232836E+00	0.23E+06	0.5177E-16	-.2098E-16	-.1176E+02	-.2480E+00	0.1762E-16	-.7067E+01	2
* 673	-0.232762E+00	0.23E+06	0.3601E-16	-.4980E-16	-.1173E+02	-.2471E+00	-.4898E-16	-.7066E+01	2
* 815	-0.204542E-01	0.20E+05	0.3731E-16	-.3498E-16	-.8442E+01	-.1947E+00	-.7169E-17	-.6284E+01	3
* 816	-0.123623E-01	0.12E+05	0.5664E-16	-.5167E-16	-.8352E+01	-.1946E+00	-.3982E-16	-.6257E+01	3
* 817	0.465015E-02	-0.47E+04	-.7081E-16	0.8067E-16	-.8165E+01	-.1945E+00	0.2490E-16	-.6200E+01	3
* 818	0.165824E-01	-0.17E+05	0.5209E-16	0.2565E-16	-.8033E+01	-.1946E+00	0.2285E-16	-.6161E+01	3
* 819	0.221251E-01	-0.22E+05	0.7540E-16	-.5478E-16	-.7972E+01	-.1947E+00	-.6859E-17	-.6142E+01	3
* 930	0.209148E+00	-0.21E+06	-.3798E-16	-.5826E-16	-.5517E+01	-.2221E+00	-.1202E-15	-.5481E+01	2
* 931	0.210123E+00	-0.21E+06	0.4768E-16	-.2723E-16	-.5496E+01	-.2226E+00	-.2655E-16	-.5477E+01	2
* 969	0.232782E+00	-0.23E+06	0.3314E-16	-.9228E-17	-.4693E+01	-.2473E+00	-.1355E-16	-.5366E+01	2
* 970	0.232847E+00	-0.23E+06	-.4531E-16	-.2432E-16	-.4671E+01	-.2482E+00	-.1155E-17	-.5365E+01	2
* 971	0.232874E+00	-0.23E+06	-.2047E-16	-.2192E-17	-.4648E+01	-.2491E+00	-.3750E-17	-.5364E+01	2
* 972	0.232861E+00	-0.23E+06	-.5596E-17	-.6167E-17	-.4626E+01	-.2500E+00	0.2581E-17	-.5363E+01	2
* 973	0.232807E+00	-0.23E+06	0.1661E-16	-.3682E-16	-.4604E+01	-.2509E+00	0.7232E-17	-.5362E+01	2
* 1021	0.167104E+00	-0.17E+06	0.7190E-18	-.2870E-16	-.3699E+01	-.3109E+00	-.5400E-17	-.5558E+01	3
* 1022	0.164629E+00	-0.16E+06	0.3760E-16	0.1252E-16	-.3688E+01	-.3122E+00	-.3747E-17	-.5567E+01	3
* 1066	0.242534E-01	-0.24E+05	0.3588E-16	0.4785E-16	-.3456E+01	-.3677E+00	-.2386E-16	-.6114E+01	3
* 1067	0.177412E-01	-0.18E+05	-.3917E-16	-.1578E-16	-.3456E+01	-.3697E+00	-.2677E-16	-.6141E+01	3
* 1068	0.984429E-02	-0.98E+04	-.5195E-17	-.6479E-16	-.3458E+01	-.3721E+00	0.5287E-16	-.6174E+01	4
* 1069	-0.107554E-02	0.11E+04	0.3507E-17	0.5043E-16	-.3462E+01	-.3753E+00	0.8465E-16	-.6221E+01	4
* 1070	-0.264425E-01	0.26E+05	-.2698E-16	0.1455E-16	-.3477E+01	-.3823E+00	0.8203E-17	-.6330E+01	4
* 1233	-0.343350E+00	0.34E+06	0.7152E-16	0.6841E-17	-.4459E+01	-.4248E+00	0.1259E-17	-.8071E+01	2
* 1234	-0.344445E+00	0.34E+06	-.4792E-16	0.1120E-17	-.4467E+01	-.4248E+00	0.1146E-16	-.8080E+01	2
* 1398	-0.435750E+00	0.44E+06	0.2257E-15	-.3707E-15	-.5898E+01	-.3780E+00	-.3254E-16	-.9464E+01	2
* 1399	-0.435761E+00	0.44E+06	-.2428E-16	0.1066E-15	-.5908E+01	-.3775E+00	-.7967E-16	-.9473E+01	2
* 1400	-0.435765E+00	0.44E+06	-.2049E-13	0.1358E-14	-.5918E+01	-.3770E+00	-.1590E-14	-.9481E+01	2
* 1401	-0.435762E+00	0.44E+06	-.2853E-15	-.2821E-15	-.5928E+01	-.3765E+00	-.3754E-16	-.9489E+01	2
* 1402	-0.435753E+00	0.44E+06	-.1598E-15	-.1811E-15	-.5938E+01	-.3760E+00	-.1115E-15	-.9497E+01	2
* 1580	-0.331597E+00	0.33E+06	-.4930E-15	0.5050E-15	-.8010E+01	-.2539E+00	-.1691E-17	-.1089E+02	2
* 1581	-0.330474E+00	0.33E+06	-.5672E-15	0.9321E-15	-.8024E+01	-.2530E+00	0.7949E-16	-.1089E+02	2
* 1789	-0.103347E-01	0.10E+05	-.7438E-16	-.7564E-16	-.1213E+02	-.1211E-01	-.2052E-15	-.1238E+02	3
* 1790	-0.726969E-02	0.73E+04	-.9706E-16	0.5381E-16	-.1221E+02	-.8720E-02	0.7074E-16	-.1240E+02	4
* 1791	-0.341595E-02	0.34E+04	-.7354E-16	-.3989E-16	-.1232E+02	-.4240E-02	-.5810E-16	-.1241E+02	4
* 1792	0.359781E-02	-0.36E+04	0.1605E-15	0.1521E-15	-.1257E+02	0.4854E-02	0.1299E-15	-.1245E+02	4
* 1793	0.642444E-02	-0.64E+04	0.5770E-16	-.1230E-15	-.1269E+02	0.9087E-02	-.1156E-15	-.1247E+02	4
* 1815	0.137843E-01	-0.14E+05	-.7873E-16	0.1205E-15	-.1337E+02	0.2796E-01	0.1533E-15	-.1253E+02	2
* 1816	0.137997E-01	-0.14E+05	-.7936E-16	-.1230E-15	-.1339E+02	0.2844E-01	0.5844E-16	-.1253E+02	2
* 1817	0.138019E-01	-0.14E+05	-.1233E-15	-.1961E-16	-.1341E+02	0.2890E-01	0.3602E-16	-.1253E+02	2
* 1818	0.137911E-01	-0.14E+05	-.5295E-16	-.8328E-16	-.1343E+02	0.2936E-01	0.7594E-17	-.1254E+02	2
* 1819	0.137672E-01	-0.14E+05	0.2722E-17	0.8255E-16	-.1346E+02	0.2981E-01	0.6709E-16	-.1254E+02	2
* 1846	0.413953E-02	-0.41E+04	0.1627E-15	0.1153E-15	-.1433E+02	0.3978E-01	-.6988E-16	-.1255E+02	3
* 1847	0.240007E-02	-0.24E+04	-.6418E-16	-.6669E-16	-.1442E+02	0.3991E-01	-.9039E-16	-.1255E+02	4
* 1848	-0.609470E-03	0.61E+03	0.2127E-16	0.1646E-15	-.1458E+02	0.3969E-01	0.6023E-16	-.1255E+02	4
* 1849	-0.155059E-01	0.16E+05	-.2793E-16	0.4145E-16	-.1578E+02	0.2102E-01	0.1394E-15	-.1247E+02	7
* 1850	-0.153784E-01	0.15E+05	0.1122E-16	-.9711E-16	-.1580E+02	0.2039E-01	0.2843E-16	-.1247E+02	3
* 1882	0.927250E-01	-0.93E+05	-.1213E-15	-.1040E-16	-.1748E+02	-.4076E-01	-.1077E-15	-.1244E+02	3
* 1883	0.971263E-01	-0.97E+05	0.5593E-16	-.3976E-16	-.1751E+02	-.4204E-01	0.9955E-16	-.1244E+02	3
* 1915	0.246890E+00	-0.25E+06	-.3316E-16	-.1103E-15	-.1835E+02	-.7499E-01	0.7449E-16	-.1254E+02	3
* 1916	0.251848E+00	-0.25E+06	-.1623E-15	-.1193E-15	-.1838E+02	-.7585E-01	-.1624E-15	-.1254E+02	3
* 1998	0.713684E+00	-0.71E+06	0.7391E-17	-.2302E-15	-.1997E+02	-.1310E+00	-.4011E-17	-.1293E+02	2
* 1999	0.719994E+00	-0.72E+06	0.1088E-15	-.3288E-16	-.1998E+02	-.1316E+00	-.7749E-16	-.1293E+02	2
* 2040	0.993136E+00	-0.99E+06	-.5285E-16	0.3851E-16	-.2068E+02	-.1532E+00	0.3018E-16	-.1315E+02	2
* 2041	0.100015E+01	-0.10E+07	-.8968E-16	0.7435E-17	-.2069E+02	-.1537E+00	0.3067E-16	-.1315E+02	2

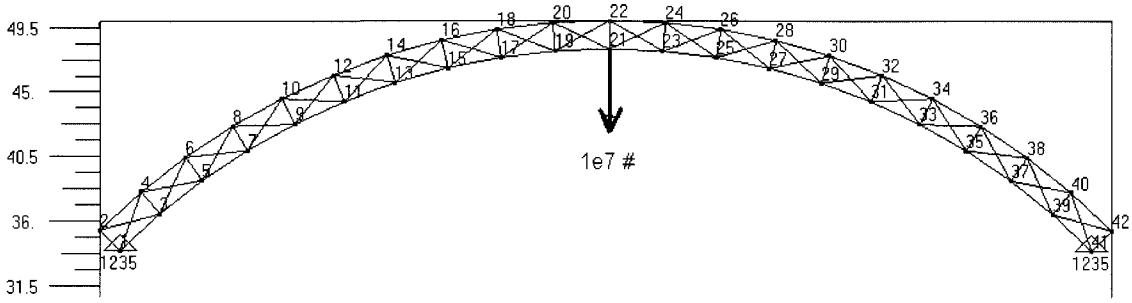
A.6.1 Star Dome Geer Example [25]





Shallow star dome model and equilibrium path of the center node from [22].

A.7 Truss Arch Example




```
*************************
Input file:
46    3   42      1    1    1   1   0  0  0  1  12  1
101   0   0   0   0   0
  6    12   24    4    4    4
1.e7    1.23   1.23   1.23   1.23   1.23   1.23   1.23   1.23   1.23   1.23
 5.d0   1.23   1.23   1.23   1.23   1.23   1.23   1.23   1.23   1.23   1.23   1.23
 1   -33.9400     33.9400   0.0
 2   -35.3500     35.3500   0.0
 3   -31.1725     36.4983   0.0
 4   -32.4675     38.0146   0.0
 5   -28.2128     38.8315   0.0
 6   -29.3848     40.4447   0.0
 7   -25.0791     40.9254   0.0
 8   -26.1210     42.6256   0.0
 9   -21.7908     42.7669   0.0
 10   -22.6961    44.5436   0.0
 11   -18.3682    44.3447   0.0
 12   -19.1313    46.1870   0.0
 13   -14.8323    45.6492   0.0
 14   -15.4485    47.5456   0.0
 15   -11.2050    46.6722   0.0
 16   -11.6705    48.6112   0.0
 17   -7.50861    47.4075   0.0
 18   -7.82054    49.3770   0.0
 19   -3.76591    47.8504   0.0
 20   -3.92236    49.8383   0.0
 21   0.000000    47.9984   0.0
 22   0.000000    49.9924   0.0
 23    3.76591    47.8504   0.0
 24    3.92236    49.8383   0.0
 25    7.50861    47.4075   0.0
 26    7.82054    49.3770   0.0
 27    11.2050    46.6722   0.0
 28    11.6705    48.6112   0.0
 29    14.8323    45.6492   0.0
 30    15.4485    47.5456   0.0
 31    18.3682    44.3447   0.0
 32    19.1313    46.1870   0.0
 33    21.7908    42.7669   0.0
 34    22.6961    44.5436   0.0
 35    25.0791    40.9254   0.0
 36    26.1210    42.6256   0.0
 37    28.2128    38.8315   0.0
 38    29.3848    40.4447   0.0
 39    31.1725    36.4983   0.0
 40    32.4675    38.0146   0.0
 41    33.9400    33.9400   0.0
 42    35.3500    35.3500   0.0
 1      1       2
 2      1       3
 3      1       4
 4      2       3
 5      2       4
```

6	3	4
7	3	5
8	3	6
9	4	5
10	4	6
11	5	6
12	5	7
13	5	8
14	6	7
15	6	8
16	7	8
17	7	9
18	7	10
19	8	9
20	8	10
21	9	10
22	9	11
23	9	12
24	10	11
25	10	12
26	11	12
27	11	13
28	11	14
29	12	13
30	12	14
31	13	14
32	13	15
33	13	16
34	14	15
35	14	16
36	15	16
37	15	17
38	15	18
39	16	17
40	16	18
41	17	18
42	17	19
43	17	20
44	18	19
45	18	20
46	19	20
47	19	21
48	19	22
49	20	21
50	20	22
51	21	22
52	21	23
53	21	24
54	22	23
55	22	24
56	23	24
57	23	25
58	23	26
59	24	25
60	24	26
61	25	26
62	25	27
63	25	28
64	26	27
65	26	28
66	27	28
67	27	29
68	27	30
69	28	29
70	28	30
71	29	30
72	29	31
73	29	32
74	30	31
75	30	32
76	31	32

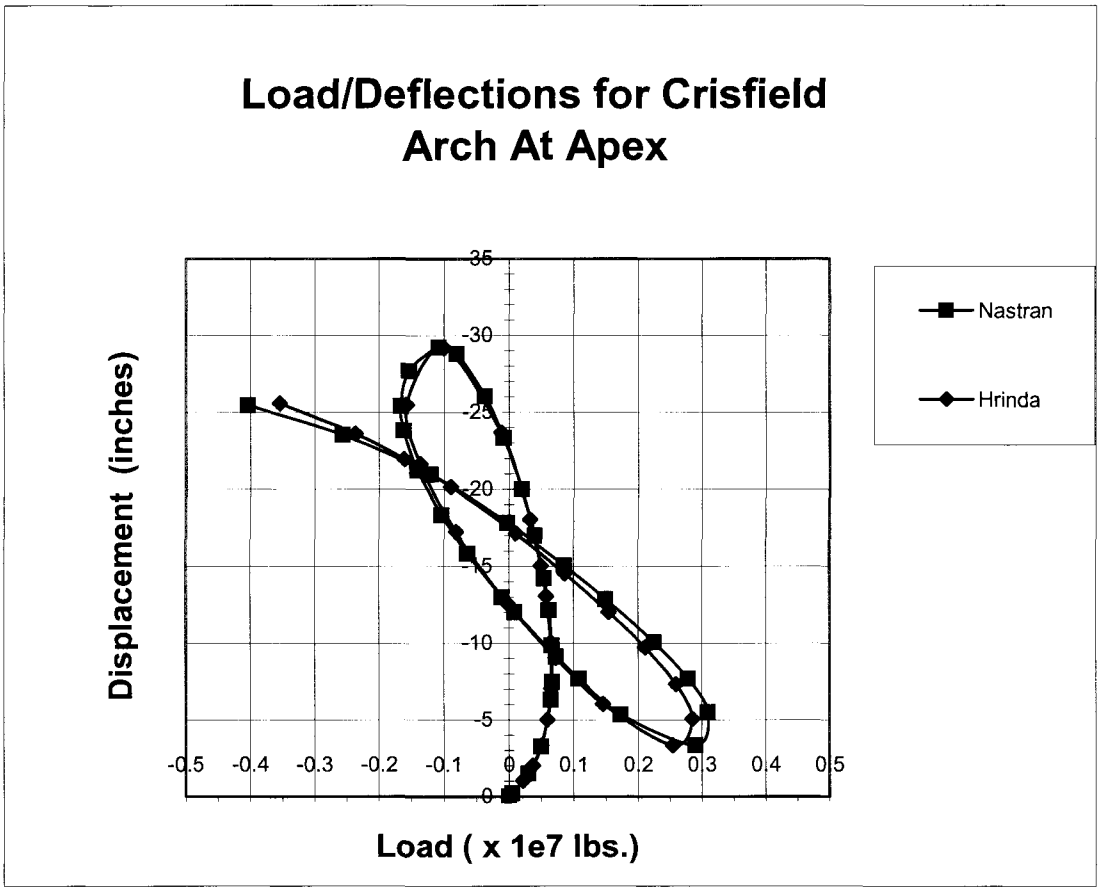
```
77    31     33
78    31     34
79    32     33
80    32     34
81    33     34
82    33     35
83    33     36
84    34     35
85    34     36
86    35     36
87    35     37
88    35     38
89    36     37
90    36     38
91    37     38
92    37     39
93    37     40
94    38     39
95    38     40
96    39     40
97    39     41
98    39     42
99    40     41
100   40     42
101   41     42
    65  -1.e7
  1  0.0
  2  0.0
  3  0.0
  6  0.0
  9  0.0
 12  0.0
 15  0.0
 18  0.0
 21  0.0
 24  0.0
 27  0.0
 30  0.0
 33  0.0
 36  0.0
 39  0.0
 42  0.0
 45  0.0
 48  0.0
 51  0.0
 54  0.0
 57  0.0
 60  0.0
 63  0.0
 66  0.0
 69  0.0
 72  0.0
 75  0.0
 78  0.0
 81  0.0
 84  0.0
 87  0.0
 90  0.0
 93  0.0
 96  0.0
 99  0.0
102  0.0
105  0.0
108  0.0
111  0.0
114  0.0
117  0.0
120  0.0
121  0.0
122  0.0
123  0.0
```

```
126  0.0

************************
```

The following are User changes made inside the program:
c....User will edit "subroutine rikarc" to suit the function
c....being solved. These are the arclength parameters that
c....produced the correct solutions.

Subroutine rikarc.....

```
itedes=1
 dl0 = .1d0
 dlmax = .1d0
  dlmin = .001d0
exlmx = 1.d0
toldis = 1.d-6
tolfor = 1.d-6
```



	istep	exlk	appliedforce	dqtot(61)	dqtot(62)	dqtot(64)	dqtot(65)	dqtot(85)	dqtot(86)	it
*	1	0.531440E-01	-0.53E+06	-.7517E-14	-.3746E+01	-.6235E-14	-.3756E+01	0.3898E+00	-.1145E+01	5
*	4	0.664279E-01	-0.66E+06	0.2493E-13	-.8209E+01	0.2064E-13	-.8233E+01	0.3418E+00	-.3228E+01	3
*	5	0.664471E-01	-0.66E+06	0.9998E-14	-.8502E+01	0.8104E-14	-.8526E+01	0.3265E+00	-.3385E+01	3
*	8	0.664154E-01	-0.66E+06	-.6406E-14	-.8697E+01	-.5501E-14	-.8722E+01	0.3156E+00	-.3490E+01	2
*	362	0.282235E-01	-0.28E+06	0.8742E-14	-.1871E+02	0.3345E-14	-.1875E+02	-.5498E+00	-.1022E+02	2
*	506	0.769218E-03	-0.77E+04	0.5957E-14	-.2228E+02	0.5431E-14	-.2231E+02	-.7713E+00	-.1331E+02	3
*	507	0.291910E-03	-0.29E+04	0.6732E-14	-.2234E+02	0.4643E-14	-.2236E+02	-.7733E+00	-.1336E+02	4
*	508	-0.869883E-03	0.87E+04	0.4835E-14	-.2247E+02	0.4446E-14	-.2250E+02	-.7778E+00	-.1348E+02	4
*	895	-0.980111E-01	0.98E+06	0.2253E-14	-.2917E+02	0.3607E-14	-.2914E+02	-.2035E-01	-.2226E+02	2
*	896	-0.985132E-01	0.99E+06	0.2510E-14	-.2917E+02	0.1652E-14	-.2914E+02	-.1406E-01	-.2228E+02	2
*	897	-0.990241E-01	0.99E+06	-.7839E-15	-.2917E+02	-.3038E-14	-.2914E+02	-.7707E-02	-.2231E+02	2
*	1011	-0.157188E+00	0.16E+07	0.8517E-13	-.2553E+02	0.1072E-12	-.2550E+02	0.4838E+00	-.2269E+02	2
*	1012	-0.157192E+00	0.16E+07	0.1082E-11	-.2550E+02	0.1324E-11	-.2546E+02	0.4834E+00	-.2268E+02	2
*	1013	-0.157191E+00	0.16E+07	0.4365E-12	-.2546E+02	0.5347E-12	-.2542E+02	0.4829E+00	-.2267E+02	2
*	1414	-0.183943E-02	0.18E+05	-.5008E-14	-.1258E+02	-.8636E-14	-.1262E+02	-.1351E+01	-.1636E+02	3
*	1415	-0.337339E-03	0.34E+04	0.6401E-14	-.1250E+02	0.5833E-14	-.1254E+02	-.1367E+01	-.1631E+02	4

* 1416	0.781457E-02	-0.78E+05	0.1009E-14	-.1209E+02	0.2602E-14	-.1213E+02	-.1451E+01	-.1601E+02	4
* 1805	0.254398E+00	-0.25E+07	-.6390E-14	-.3340E+01	-.5319E-14	-.3356E+01	-.2244E+01	-.6707E+01	2
* 1806	0.255460E+00	-0.26E+07	-.3141E-14	-.3340E+01	-.2339E-14	-.3356E+01	-.2236E+01	-.6676E+01	2
* 1807	0.256540E+00	-0.26E+07	-.9737E-15	-.3341E+01	-.1137E-14	-.3358E+01	-.2227E+01	-.6645E+01	2
* 1844	0.285894E+00	-0.29E+07	-.1057E-13	-.4609E+01	-.8906E-14	-.4647E+01	-.1822E+01	-.6133E+01	2
* 1845	0.285911E+00	-0.29E+07	-.1826E-12	-.4652E+01	-.1647E-12	-.4690E+01	-.1815E+01	-.6140E+01	2
* 1846	0.285906E+00	-0.29E+07	-.2261E-13	-.4695E+01	-.1896E-13	-.4734E+01	-.1809E+01	-.6147E+01	2
* 2218	0.287532E-02	-0.29E+05	0.2494E-14	-.1739E+02	0.5450E-14	-.1754E+02	-.1431E+01	-.1180E+02	3
* 2219	0.220860E-03	-0.22E+04	-.1007E-13	-.1747E+02	-.7334E-14	-.1762E+02	-.1434E+01	-.1185E+02	4
* 2220	-0.257602E-01	0.26E+06	0.2591E-14	-.1830E+02	-.4778E-15	-.1844E+02	-.1455E+01	-.1230E+02	5
* 2630	-0.306386E+00	0.31E+07	-.9173E-14	-.2184E+02	-.1090E-13	-.2177E+02	-.1099E+01	-.1626E+02	2

A.7.1 Typical arch equilibrium paths ([29], page 109)

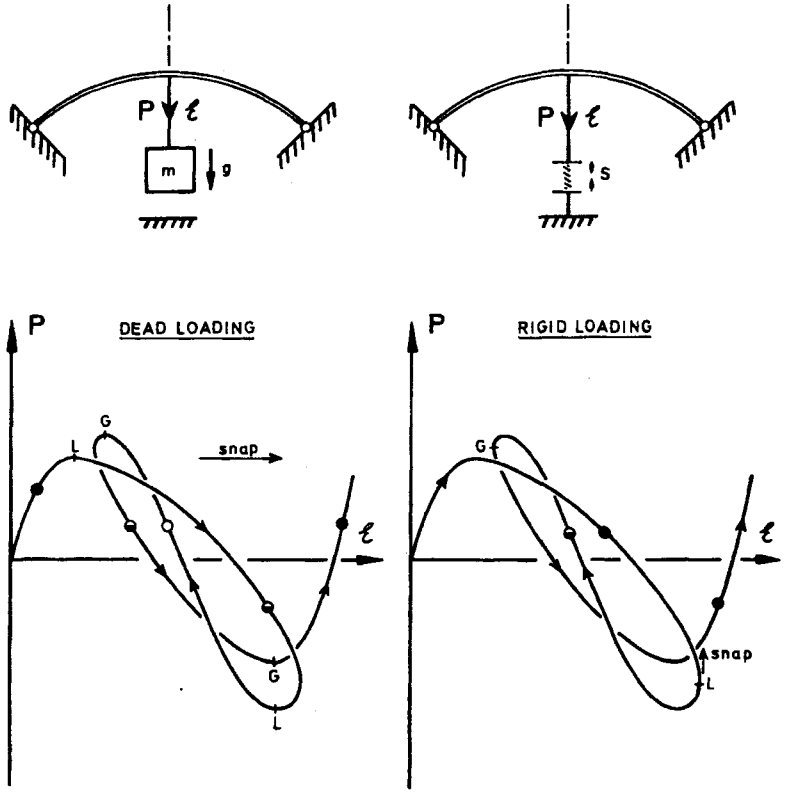


Figure 5.6 Load–corresponding deflection response of a symmetric arch (after Biexeno and Grammel, 1960). A solid circle denotes zero, a half-solid circle denotes one, and an open circle denotes two degrees of instability. Note that ℓ here represents the corresponding deflection of the load (not the end-restraint, as in the rest of the chapter)

APPENDIX B- Hrinda Rikarc Optimization Users' Input Data Description

B.1 Optimum Khot 2 Element Symmetric Example

Two-element symmetric truss

The input files:

khot_2ele_mod_in_sym.dat

```
7 3 3 1  1  1  2  0  0  0  1  12  1  1
2  0  0  0  0  0
 6  12  24  4  4  4
1.e7  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1  0.     0.    0.
2  125.   2.5    0.
3  250.   0.    0.
  1  1  1  2
  2  2  2  3
5 -200.d0
5  1  -2.5
```

khot_2ele_mod_in_sym_areas.dat

```
3.0d0
5.0d0
```

khot_2ele_mod_sym_sup.dat

```
 1  0.0
 2  0.0
 3  0.0
 6  0.0
 7  0.0
 8  0.0
 9  0.0
```

```
            open(unit=5,file='khot_2ele_mod_in_sym.dat',status='old')
            open(unit=8,file='khot_2ele_mod_in_sym_areas.dat',status='old')

            open(unit=5,file='khot_2ele_mod_in_sym.dat',status='old')

            open(unit=51,file='khot_2ele_mod_sym_sup.dat',status='old')

            dSEDtol=1.d-3
            Amin=.1d0
C** Shallow_truss & Khot 4element
      itedes=1
      dl0 = .1d0
      dlmax = .10d0
      dlmin = .001d0
      exlmx = 1.d0
      toldis = 1.d-4
      tolfor = 1.d-4

c** "Khot_2ele_mod_in.dat"**************************************************
      appliedforce = exlk * exfor(5)
      weightotal=0.d0
            Do 55 i=1,nel
            weightotal=weightotal+weight(i)
55          Continue

     if(istep .eq. 1) then
c             write(15,63)'N =',N
c 63   format(i4)
     write(15,53)
 53   format('  istep     exlk      appliedforce      a(4)
    $  a(5)    it      Weight     Area1       Area2
    $        xlie 1     xlie 2    fin1   N  Nevenodd')
            endif
```

```
c           If(Nevenodd .eq. 2 .or. Nevenodd .eq. 0)then

     write(15,50) '*',istep,exlk,appliedforce,
    c   dqtot(4),dqtot(5),it,weightotal,
    c   propsect(1),propsect(2),xlie(1),xlie(2),finI(1),N,
    c   Nevenodd

c           endif

50   format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
    c 1x,i5,1x,e14.6,2x,e12.6,2x,e12.6,2x,e14.8,2X,e14.8,1x,e8.2,
    c  i4,2x,i4)
```

MSC-Nastran run deck
Khot_2ele_sym_msc_2009.dat

```
ID E:\Mscn4w30\data\PhD\Opti,FEMAP
SOL NLSTATIC
TIME 10000
CEND
 ECHO = NONE
 DISPLACEMENT = ALL
 SPC = 1
 LOAD = 1
 NLPARM = 1
BEGIN BULK
$ ******************************************************************************
$  Written by : FEMAP
$  Version    : 5.00
$  Translator : MSC/NASTRAN
$  From Model : E:\Mscn4w30\data\PhD\Optimization\Khot_2ele_GAH_symmetric.MOD
$  Date       : Sun Feb 02 18:09:01 2003
$ ******************************************************************************
$
NLPARM       1    200         AUTO      5    200     UP     ALL+
+         0.01   0.01  1.E-7    30     25     4    0.2    0.5+
+            5                  20.           20.
NLPCI        1  MRIKS   0.25     4.                12     900
PARAM,LGDISP,1
PARAM,AUTOSPC,YES
PARAM,GRDPNT,0
CORD2C       1     0    0.    0.    0.    0.    0.    1.+FEMAPC1
+FEMAPC1     1.    0.    1.
CORD2S       2     0    0.    0.    0.    0.    0.    1.+FEMAPC2
+FEMAPC2     1.    0.    1.
$ FEMAP Load Set 1 : Untitled
FORCE        1     2     0     1.    0.  -200.     0.
$ FEMAP Constraint Set 1 : Untitled
SPC          1     1  12345    0.
SPC          1     2    345    0.
SPC          1     3  12345    0.
$ FEMAP Property 1 : Untitled
PROD         1     1  6.4999    0.    0.    0.
$ FEMAP Property 2 : Untitled
PROD         2     1  6.4999    0.    0.    0.
$ FEMAP Material 1 : Untitled
MAT1         1  1.E+7                0.    0.    0.
GRID         1     0    0.    0.    0.     0
GRID         2     0   125.   2.5    0.     0
GRID         3     0   250.   0.    0.     0
CROD         1     1     1     2
CROD         2     2     2     3
ENDDATA
```

SED_out.txt

```
 1SED= 0.138806E+01 Area =  0.300000E+01 exlk=  0.577049E+00 SSP=  0.000000E+00 SEDavg=  0.943830E+00 N=  1
 2SED= 0.499600E+00 Area =  0.500000E+01 exlk=  0.577049E+00 SSP=  0.000000E+00 SEDavg=  0.943830E+00 N=  1

 1SED= 0.820330E+00 Area =  0.333916E+01 exlk=  0.493552E+00 SSP=  0.577049E+00 SEDavg=  0.890393E+00 N=  2
```

```
2SED= 0.960456E+00 Area = 0.308603E+01 exlk= 0.493552E+00 SSP= 0.577049E+00 SEDavg= 0.890393E+00 N= 2
_____
1SED= 0.887714E+00 Area = 0.319704E+01 exlk= 0.491683E+00 SSP= 0.493551E+00 SEDavg= 0.888604E+00 N= 3
2SED= 0.889494E+00 Area = 0.319384E+01 exlk= 0.491683E+00 SSP= 0.493551E+00 SEDavg= 0.888604E+00 N= 3
_____
1SED= 0.888589E+00 Area = 0.319546E+01 exlk= 0.491682E+00 SSP= 0.491682E+00 SEDavg= 0.888604E+00 N= 4
2SED= 0.888619E+00 Area = 0.319541E+01 exlk= 0.491682E+00 SSP= 0.491682E+00 SEDavg= 0.888604E+00 N= 4
_____
1SED= 0.877988E+00 Area = 0.649904E+01 exlk= 0.100000E+01 SSP= 0.491682E+00 SEDavg= 0.878002E+00 N= 5
2SED= 0.878017E+00 Area = 0.649893E+01 exlk= 0.100000E+01 SSP= 0.491682E+00 SEDavg= 0.878002E+00 N= 5
_____
```

fea3d2_out.txt

	istep	exlk	appliedforce	a(4)	a(5)	it	Weight	Area1	Area2	xlie 1	xlie 2	fin1	N	Nevenodd
*	1	0.100000E+00	-0.20E+02	0.4083-319	0.5105-316	3	0.162507E+03	0.649904E+01	0.649893E+01	0.12502402E+03	0.12502402E+03	-.51E+03	5	2
*	2	0.168959E+00	-0.34E+02	0.8661E-08	-.5319E-01	3	0.162507E+03	0.649904E+01	0.649893E+01	0.12502331E+03	0.12502331E+03	-.87E+03	5	2
*	25	0.257597E+00	-0.52E+02	0.2150E-07	-.1342E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12502238E+03	0.12502238E+03	-.14E+04	5	2
*	103	0.329470E+00	-0.66E+02	0.2802E-07	-.1765E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12502159E+03	0.12502159E+03	-.18E+04	5	2
*	188	0.405717E+00	-0.81E+02	0.3525E-07	-.2242E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12502071E+03	0.12502071E+03	-.22E+04	5	2
*	310	0.510999E+00	-0.10E+03	0.4587E-07	-.2963E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12501942E+03	0.12501942E+03	-.29E+04	5	2
*	467	0.638187E+00	-0.13E+03	0.6003E-07	-.3961E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12501770E+03	0.12501770E+03	-.38E+04	5	2
*	571	0.716302E+00	-0.14E+03	0.6976E-07	-.4676E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12501652E+03	0.12501652E+03	-.44E+04	5	2
*	708	0.809851E+00	-0.16E+03	0.8304E-07	-.5693E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12501491E+03	0.12501491E+03	-.52E+04	5	2
*	874	0.904407E+00	-0.18E+03	0.9981E-07	-.7061E+00	2	0.162507E+03	0.649904E+01	0.649893E+01	0.12501287E+03	0.12501287E+03	-.63E+04	5	2
*	1232	0.999990E+00	-0.20E+03	0.1363E-06	-.1047E+01	1	0.162507E+03	0.649904E+01	0.649893E+01	0.12500844E+03	0.12500844E+03	-.86E+04	5	2
*	1233	0.100000E+01	-0.20E+03	0.1364E-06	-.1048E+01	1	0.162507E+03	0.649904E+01	0.649893E+01	0.12500843E+03	0.12500843E+03	-.86E+04	5	2

B.2 Optimum Khot 2 Element Unsymmetric Example

Two-element asymmetric truss

The input files:

khot_2ele_mod_in_unsym.dat

```
7 3 3 1  1  1 2  0 0 0  1  12  1  1
2 0 0 0 0 0
 6  12 24  4  4  4
1.e7  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1  0.     0.    0.
2  200.   2.5    0.
3  250.   0.    0.
  1  2 1  2
  2  1 2  3
5 -200.d0
5  1  -2.5
```

khot_2ele_mod_in_unsym_areas.dat

```
1.0d0
2.0d0
```

khot_2ele_mod_unsym_sup.dat

```
 1  0.0
 2  0.0
 3  0.0
 6  0.0
 7  0.0
 8  0.0
 9  0.0
```

```
open(unit=5,file='khot_2ele_mod_in_unsym.dat',status='old')
```

```
            open(unit=8,file='khot_2ele_mod_in_unsym_areas.dat',status='old')

            open(unit=5,file='khot_2ele_mod_in_unsym.dat',status='old')

            open(unit=51,file='khot_2ele_mod_unsym_sup.dat',status='old')

            dSEDtol=1.d-2 !Khot_2ele
C**  Khot paper
      itedes=1
      dl0 = .1d0
      dlmax = .10d0
      dlmin = .001d0
      exlmx = 1.d0
      toldis = 1.d-4
      tolfor = 1.d-4

c** "Khot_2ele_mod_in.dat"************************************************
      appliedforce = exlk * exfor(5)
      weightotal=0.d0
            Do 55 i=1,nel
c           print *,'istep=',istep
c           print *,'i=',i
c           print *,'weightotal=',weightotal
c           print *,'weight(_)=',weight(i)
c           pause
cd    SEDratio(i)=strainD(i)/weight(i)
            weightotal=weightotal+weight(i)
55          Continue

     if(istep .eq. 1) then
c            write(15,63)'N =',N
c 63   format(i4)
     write(15,53)
 53  format('  istep     exlk       appliedforce       a(4)
    $ a(5)    it       Weight      Area1        Area2
    $        xlie 1      xlie 2     fin1   N  Nevenodd')
            endif

c           If(Nevenodd .eq. 2 .or. Nevenodd .eq. 0)then

     write(15,50) '*',istep,exlk,appliedforce,
    c   dqtot(4),dqtot(5),it,weightotal,
    c   propsect(1),propsect(2),xlie(1),xlie(2),fin1(1),N,
    c   Nevenodd

c           endif

50   format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
    c 1x,i5,1x,e14.6,2x,e12.6,2x,e12.6,2x,e14.8,2X,e14.8,1x,e8.2,
    c  i4,2x,i4)
```

MSC-Nastran run deck
Khot_2bar_unsym_msc_2009.dat

```
INIT MASTER(S)
ID E:\Mscn4w30\data\PhD\Opti,MSC/N
SOL NLSTATIC
TIME 10000
DIAG 5
CEND
 ECHO = NONE
 DISPLACEMENT = ALL
 SPC = 1
 LOAD = 1
 NLPARM = 1
BEGIN BULK
$ ******************************************************************************
$  Written by : MSC/NASTRAN for Windows
$  Version    : 5.00
$  Translator : MSC/NASTRAN
$  From Model : E:\Mscn4w30\data\PhD\Optimization\Khot_2.MOD
```

```
$  Date     : Wed Jan 29 18:55:33 2003
$ ************************************************************************
$
NLPARM       1    500         AUTO     5     5     UP    ALL+
+         0.001  0.001  1.E-7    30     25     4     0.2    0.5+
+           5                   20.          20.
NLPCI        1  MRIKS   0.25     4.                200    500
PARAM,LGDISP,1
PARAM,PRGPST,NO
PARAM,AUTOSPC,YES
PARAM,K6ROT,100.
PARAM,MAXRATIO,1.E+8
PARAM,GRDPNT,0
CORD2C       1     0     0.    0.    0.    0.    0.    1.+MSC/NC1
+MSC/NC1     1.    0.    1.
CORD2S       2     0     0.    0.    0.    0.    0.    1.+MSC/NC2
+MSC/NC2     1.    0.    1.
$ MSC/NASTRAN for Windows Load Set 1 : Untitled
FORCE        1     2     0     1.    0.  -200.     0.
$ MSC/NASTRAN for Windows Constraint Set 1 : Untitled
SPC          1     1   12345     0.
SPC          1     2     345     0.
SPC          1     3   12345     0.
$ MSC/NASTRAN for Windows Property 1 : Untitled
PROD         1     1 2.66409     0.    0.    0.
$ MSC/NASTRAN for Windows Property 2 : Untitled
PROD         2     1  2.6673     0.    0.    0.
$ MSC/NASTRAN for Windows Material 1 : Untitled
MAT1         1   1.E+7                 0.    0.    0.
GRID         1     0     0.    0.    0.     0
GRID         2     0   200.    2.5   0.     0
GRID         3     0   250.    0.    0.     0
CROD         1     1     1     2
CROD         2     2     2     3
ENDDATA
```

SED_out.tx

```
 1SED= 0.150470E+01 Area =  0.200000E+01 exlk=  0.625349E+00 SSP=  0.000000E+00 SEDavg=  0.376565E+01 N=  1
 2SED= 0.602659E+01 Area =  0.100000E+01 exlk=  0.625349E+00 SSP=  0.000000E+00 SEDavg=  0.376565E+01 N=  1
 ------------------------------------------------------------------------------------------------------------
 1SED= 0.231180E+01 Area =  0.929589E+00 exlk=  0.360397E+00 SSP=  0.625348E+00 SEDavg=  0.197169E+01 N=  2
 2SED= 0.163158E+01 Area =  0.110690E+01 exlk=  0.360397E+00 SSP=  0.625348E+00 SEDavg=  0.197169E+01 N=  2
 ------------------------------------------------------------------------------------------------------------
 1SED= 0.221038E+01 Area =  0.974113E+00 exlk=  0.369088E+00 SSP=  0.360397E+00 SEDavg=  0.210798E+01 N=  3
 2SED= 0.200558E+01 Area =  0.102302E+01 exlk=  0.369088E+00 SSP=  0.360397E+00 SEDavg=  0.210798E+01 N=  3
 ------------------------------------------------------------------------------------------------------------
 1SED= 0.217852E+01 Area =  0.990456E+00 exlk=  0.372661E+00 SSP=  0.369087E+00 SEDavg=  0.215116E+01 N=  4
 2SED= 0.212381E+01 Area =  0.100352E+01 exlk=  0.372661E+00 SSP=  0.369087E+00 SEDavg=  0.215116E+01 N=  4
 ------------------------------------------------------------------------------------------------------------
 1SED= 0.217041E+01 Area =  0.995071E+00 exlk=  0.373692E+00 SSP=  0.372660E+00 SEDavg=  0.216341E+01 N=  5
 2SED= 0.215641E+01 Area =  0.998681E+00 exlk=  0.373692E+00 SSP=  0.372660E+00 SEDavg=  0.216341E+01 N=  5
 ------------------------------------------------------------------------------------------------------------
 1SED= 0.216756E+01 Area =  0.996269E+00 exlk=  0.373962E+00 SSP=  0.373691E+00 SEDavg=  0.216579E+01 N=  6
 2SED= 0.216402E+01 Area =  0.997468E+00 exlk=  0.373962E+00 SSP=  0.373691E+00 SEDavg=  0.216579E+01 N=  6
 ------------------------------------------------------------------------------------------------------------
 1SED= 0.215320E+01 Area =  0.266409E+01 exlk=  0.100000E+01 SSP=  0.373691E+00 SEDavg=  0.215135E+01 N=  7
 2SED= 0.214950E+01 Area =  0.266730E+01 exlk=  0.100000E+01 SSP=  0.373691E+00 SEDavg=  0.215135E+01 N=  7
 ------------------------------------------------------------------------------------------------------------
```

fea3d2_out.txt

```
  istep      exlk       appliedforce    a(4)        a(5)        it   Weight          Area1          Area2          xlie 1          xlie 2          fin1  N  Nevenodd
*   1  0.100000E+00   -0.20E+02  0.2119-317   0.5484-316   3  0.666391E+02 0.266730E+01 0.266409E+01 0.20001317E+03 0.50061848E+02 -.33E+03  7   2
*   2  0.168959E+00   -0.34E+02  -.1971E-02   -.5315E-01   3  0.666391E+02 0.266730E+01 0.266409E+01 0.20001142E+03 0.50061409E+02 -.56E+03  7   2
*   3  0.208033E+00   -0.42E+02  -.3286E-02   -.8926E-01   3  0.666391E+02 0.266730E+01 0.266409E+01 0.20001040E+03 0.50061154E+02 -.70E+03  7   2

* 100  0.326738E+00   -0.65E+02  -.6319E-02   -.1747E+00   2  0.666391E+02 0.266730E+01 0.266409E+01 0.20000718E+03 0.50060348E+02 -.11E+04  7   2

* 663  0.780432E+00   -0.16E+03  -.1789E-01   -.5346E+00   2  0.666391E+02 0.266730E+01 0.266409E+01 0.19999176E+03 0.50056491E+02 -.32E+04  7   2
```

```
* 1233  0.999964E+00  -0.20E+03  -.3104E-01  -.1048E+01  1  0.666391E+02 0.266730E+01 0.266409E+01 0.19997423E+03 0.50052110E+02 -.55E+04  7  2
* 1234  0.999976E+00  -0.20E+03  -.3106E-01  -.1049E+01  1  0.666391E+02 0.266730E+01 0.266409E+01 0.19997420E+03 0.50052103E+02 -.55E+04  7  2
* 1235  0.999987E+00  -0.20E+03  -.3108E-01  -.1050E+01  1  0.666391E+02 0.266730E+01 0.266409E+01 0.19997418E+03 0.50052095E+02 -.55E+04  7  2
* 1236  0.999996E+00  -0.20E+03  -.3110E-01  -.1051E+01  1  0.666391E+02 0.266730E+01 0.266409E+01 0.19997415E+03 0.50052088E+02 -.55E+04  7  2
* 1237  0.100000E+01  -0.20E+03  -.3113E-01  -.1052E+01  1  0.666391E+02 0.266730E+01 0.266409E+01 0.19997412E+03 0.50052081E+02 -.55E+04  7  2
```

B.3 Optimum Khot 4 Element Asymmetric Example

four-member asymmetric

The input files:

khot_4ele_mod_in.dat

```
12 3 5 1  1  1  4  0  0  0  1  12  1  1
4  0  0  0  0  0
 6  12  24  4  4  4
1.e7  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
1   120.  -60.    0.
2   120.  144.    0.
3   -72.  144.    0.
4   -72.  -60.    0.
5    0.    0.    3.
  1  1  1  5
  2  2  2  5
  3  3  3  5
  4  4  4  5
15 -200.d0
15  1  1.0
```

khot_4ele_mod_in_areas.dat

```
 2.20d0
 2.20d0
 2.20d0
 2.20d0
```

khot_4ele_mod_sup.dat

```
 1  0.0
 2  0.0
 3  0.0
 4  0.0
 5  0.0
 6  0.0
 7  0.0
 8  0.0
 9  0.0
10  0.0
11  0.0
12  0.0
```

```
            open(unit=5,file='khot_4ele_mod_in.dat',status='old')
            open(unit=8,file='khot_4ele_mod_in_areas.dat',status='old')

            open(unit=5,file='khot_4ele_mod_in.dat',status='old')

            dSEDtol=1.d-2 !Khot_4ele
            Amin=.1d0

C** Shallow_truss & Khot 4element
      itedes=1
      dl0 = .1d0
      dlmax = .10d0
      dlmin = .001d0
      exlmx = 1.d0
      toldis = 1.d-4
      tolfor = 1.d-4

c**"Khot_4ele_mod_in_initial.dat"*********************************************
      appliedforce = exlk * exfor(15)
```

```
      weightotal=0.d0
            Do 55 i=1,nel
            weightotal=weightotal+weight(i)
55          Continue

     if(istep .eq. 1) then
     write(15,53)
 53  format(' istep     exlk      appliedforce      a(14)
    $ a(15)    it       Weight     Area1       Area2
    $ Area3       Area4     xlie 1      xlie 2     fin1')
            endif

     write(15,50) '*',istep,exlk,appliedforce,
    c   dqtot(14),dqtot(15),it,weightotal,
    c   propsect(1),propsect(2),propsect(3),propsect(4),
    c   xlie(1),xlie(2),fin1(1)

50   format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
    c 1x,i5,1x,e14.8,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
    c  e14.8,2X,e14.8,1x,e8.2)
```

MSC-Nastran run deck
Khot_4bar_msc_2009.dat

```
INIT MASTER(S)
ID E:\Mscn4w30\data\PhD\Opti,MSC/N
SOL SESTATICS
TIME 10000
DIAG 5
CEND
 ECHO = NONE
 DISPLACEMENT = ALL
 SPC = 1
 LOAD = 1
BEGIN BULK
$ ******************************************************************************
$  Written by : MSC/NASTRAN for Windows
$  Version    : 5.00
$  Translator : MSC/NASTRAN
$  From Model : E:\Mscn4w30\data\PhD\Optimization\Khot_4bar_2009.MOD
$  Date       : Wed Jan 29 18:23:28 2003
$ ******************************************************************************
$
PARAM,PRGPST,NO
PARAM,AUTOSPC,YES
PARAM,K6ROT,100.
PARAM,MAXRATIO,1.E+8
PARAM,GRDPNT,0
CORD2C        1       0     0.      0.      0.      0.      0.      1.+MSC/NC1
+MSC/NC1      1.      0.      1.
CORD2S        2       0     0.      0.      0.      0.      0.      1.+MSC/NC2
+MSC/NC2      1.      0.      1.
$ MSC/NASTRAN for Windows Load Set 1 : Untitled
FORCE         1       5     0       1.      0.      0.   -200.
$ MSC/NASTRAN for Windows Constraint Set 1 : Untitled
SPC           1       1     123     0.
SPC           1       2     123     0.
SPC           1       3     123     0.
SPC           1       4     123     0.
$ MSC/NASTRAN for Windows Property 1 : Untitled
PROD          1       1  2.1724     0.      0.      0.
$ MSC/NASTRAN for Windows Property 2 : Untitled
PROD          2       1 1.63832     0.      0.      0.
$ MSC/NASTRAN for Windows Property 3 : Untitled
PROD          3       1 1.72547     0.      0.      0.
$ MSC/NASTRAN for Windows Property 4 : Untitled
PROD          4       1  2.9233     0.      0.      0.
$ MSC/NASTRAN for Windows Material 1 : Untitled
MAT1          1  1.E+7            0.1     0.1     0.      0.
GRID          1       0   120.   -60.     0.      0
GRID          2       0   120.   144.     0.      0
```

```
GRID       3    0  -72.   144.    0.     0
GRID       4    0  -72.   -60.    0.     0
GRID       5    0   0.     0.     3.     0
CROD       1    1    1     5
CROD       2    2    2     5
CROD       3    3    3     5
CROD       4    4    4     5
ENDDATA
```

SED_out.txt

```
1SED= 0.105498E+01 Area = 0.220000E+01 exlk= 0.100030E+01 SSP= 0.000000E+00 SEDavg= 0.105988E+01 N=  1
2SED= 0.620698E+00 Area = 0.220000E+01 exlk= 0.100030E+01 SSP= 0.000000E+00 SEDavg= 0.105988E+01 N=  1
3SED= 0.673578E+00 Area = 0.220000E+01 exlk= 0.100030E+01 SSP= 0.000000E+00 SEDavg= 0.105988E+01 N=  1
4SED= 0.189025E+01 Area = 0.220000E+01 exlk= 0.100030E+01 SSP= 0.000000E+00 SEDavg= 0.105988E+01 N=  1

1SED= 0.118837E+01 Area = 0.215400E+01 exlk= 0.100013E+01 SSP= 0.100000E+00 SEDavg= 0.117587E+01 N=  2
2SED= 0.783540E+00 Area = 0.204272E+01 exlk= 0.100013E+01 SSP= 0.100000E+00 SEDavg= 0.117587E+01 N=  2
3SED= 0.828873E+00 Area = 0.205949E+01 exlk= 0.100013E+01 SSP= 0.100000E+00 SEDavg= 0.117587E+01 N=  2
4SED= 0.190271E+01 Area = 0.228335E+01 exlk= 0.100013E+01 SSP= 0.100000E+00 SEDavg= 0.117587E+01 N=  2

1SED= 0.132184E+01 Area = 0.212725E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.129867E+01 N=  3
2SED= 0.951726E+00 Area = 0.193505E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.129867E+01 N=  3
3SED= 0.990069E+00 Area = 0.196194E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.129867E+01 N=  3
4SED= 0.193105E+01 Area = 0.236368E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.129867E+01 N=  3

1SED= 0.152666E+01 Area = 0.211247E+01 exlk= 0.100001E+01 SSP= 0.100000E+00 SEDavg= 0.149635E+01 N=  4
2SED= 0.117723E+01 Area = 0.185951E+01 exlk= 0.100001E+01 SSP= 0.100000E+00 SEDavg= 0.149635E+01 N=  4
3SED= 0.121097E+01 Area = 0.189281E+01 exlk= 0.100001E+01 SSP= 0.100000E+00 SEDavg= 0.149635E+01 N=  4
4SED= 0.207052E+01 Area = 0.243794E+01 exlk= 0.100001E+01 SSP= 0.100000E+00 SEDavg= 0.149635E+01 N=  4

1SED= 0.156036E+01 Area = 0.210494E+01 exlk= 0.993698E+00 SSP= 0.100000E+00 SEDavg= 0.152974E+01 N=  5
2SED= 0.126967E+01 Area = 0.180534E+01 exlk= 0.993698E+00 SSP= 0.100000E+00 SEDavg= 0.152974E+01 N=  5
3SED= 0.129604E+01 Area = 0.184287E+01 exlk= 0.993698E+00 SSP= 0.100000E+00 SEDavg= 0.152974E+01 N=  5
4SED= 0.199289E+01 Area = 0.250440E+01 exlk= 0.993698E+00 SSP= 0.100000E+00 SEDavg= 0.152974E+01 N=  5

1SED= 0.143814E+01 Area = 0.207251E+01 exlk= 0.937731E+00 SSP= 0.993697E+00 SEDavg= 0.150872E+01 N=  6
2SED= 0.174300E+01 Area = 0.144827E+01 exlk= 0.937731E+00 SSP= 0.993697E+00 SEDavg= 0.150872E+01 N=  6
3SED= 0.173137E+01 Area = 0.150888E+01 exlk= 0.937731E+00 SSP= 0.993697E+00 SEDavg= 0.150872E+01 N=  6
4SED= 0.112235E+01 Area = 0.314450E+01 exlk= 0.937731E+00 SSP= 0.993697E+00 SEDavg= 0.150872E+01 N=  6

1SED= 0.155377E+01 Area = 0.193008E+01 exlk= 0.905957E+00 SSP= 0.937730E+00 SEDavg= 0.152453E+01 N=  7
2SED= 0.130065E+01 Area = 0.161519E+01 exlk= 0.905957E+00 SSP= 0.937730E+00 SEDavg= 0.152453E+01 N=  7
3SED= 0.132268E+01 Area = 0.167226E+01 exlk= 0.905957E+00 SSP= 0.937730E+00 SEDavg= 0.152453E+01 N=  7
4SED= 0.192100E+01 Area = 0.232092E+01 exlk= 0.905957E+00 SSP= 0.937730E+00 SEDavg= 0.152453E+01 N=  7

1SED= 0.145933E+01 Area = 0.191715E+01 exlk= 0.871200E+00 SSP= 0.905956E+00 SEDavg= 0.150599E+01 N=  8
2SED= 0.167527E+01 Area = 0.136565E+01 exlk= 0.871200E+00 SSP= 0.905956E+00 SEDavg= 0.150599E+01 N=  8
3SED= 0.166541E+01 Area = 0.143558E+01 exlk= 0.871200E+00 SSP= 0.905956E+00 SEDavg= 0.150599E+01 N=  8
4SED= 0.122396E+01 Area = 0.279394E+01 exlk= 0.871200E+00 SSP= 0.905956E+00 SEDavg= 0.150599E+01 N=  8

1SED= 0.153643E+01 Area = 0.184131E+01 exlk= 0.858015E+00 SSP= 0.871199E+00 SEDavg= 0.151474E+01 N=  9
2SED= 0.138153E+01 Area = 0.147919E+01 exlk= 0.858015E+00 SSP= 0.871199E+00 SEDavg= 0.151474E+01 N=  9
3SED= 0.139362E+01 Area = 0.154695E+01 exlk= 0.858015E+00 SSP= 0.871199E+00 SEDavg= 0.151474E+01 N=  9
4SED= 0.174739E+01 Area = 0.230218E+01 exlk= 0.858015E+00 SSP= 0.871199E+00 SEDavg= 0.151474E+01 N=  9

1SED= 0.148227E+01 Area = 0.184888E+01 exlk= 0.845865E+00 SSP= 0.858015E+00 SEDavg= 0.150482E+01 N= 10
2SED= 0.159644E+01 Area = 0.135583E+01 exlk= 0.845865E+00 SSP= 0.858015E+00 SEDavg= 0.150482E+01 N= 10
3SED= 0.159025E+01 Area = 0.142857E+01 exlk= 0.845865E+00 SSP= 0.858015E+00 SEDavg= 0.150482E+01 N= 10
4SED= 0.135030E+01 Area = 0.258145E+01 exlk= 0.845865E+00 SSP= 0.858015E+00 SEDavg= 0.150482E+01 N= 10

1SED= 0.152101E+01 Area = 0.181862E+01 exlk= 0.842990E+00 SSP= 0.845865E+00 SEDavg= 0.150874E+01 N= 11
2SED= 0.144344E+01 Area = 0.142002E+01 exlk= 0.842990E+00 SSP= 0.845865E+00 SEDavg= 0.150874E+01 N= 11
3SED= 0.144893E+01 Area = 0.149131E+01 exlk= 0.842990E+00 SSP= 0.845865E+00 SEDavg= 0.150874E+01 N= 11
4SED= 0.162158E+01 Area = 0.234661E+01 exlk= 0.842990E+00 SSP= 0.845865E+00 SEDavg= 0.150874E+01 N= 11

1SED= 0.149566E+01 Area = 0.182757E+01 exlk= 0.839536E+00 SSP= 0.842989E+00 SEDavg= 0.150572E+01 N= 12
2SED= 0.154970E+01 Area = 0.136541E+01 exlk= 0.839536E+00 SSP= 0.842989E+00 SEDavg= 0.150572E+01 N= 12
3SED= 0.154651E+01 Area = 0.143855E+01 exlk= 0.839536E+00 SSP= 0.842989E+00 SEDavg= 0.150572E+01 N= 12
4SED= 0.143100E+01 Area = 0.248895E+01 exlk= 0.839536E+00 SSP= 0.842989E+00 SEDavg= 0.150572E+01 N= 12
```

1SED= 0.151299E+01 Area = 0.181575E+01 exlk= 0.839265E+00 SSP= 0.839536E+00 SEDavg= 0.150689E+01 N= 13
2SED= 0.147660E+01 Area = 0.139762E+01 exlk= 0.839265E+00 SSP= 0.839536E+00 SEDavg= 0.150689E+01 N= 13
3SED= 0.147902E+01 Area = 0.146994E+01 exlk= 0.839265E+00 SSP= 0.839536E+00 SEDavg= 0.150689E+01 N= 13
4SED= 0.155894E+01 Area = 0.238279E+01 exlk= 0.839265E+00 SSP= 0.839536E+00 SEDavg= 0.150689E+01 N= 13

1SED= 0.150188E+01 Area = 0.182117E+01 exlk= 0.838257E+00 SSP= 0.839265E+00 SEDavg= 0.150638E+01 N= 14
2SED= 0.152678E+01 Area = 0.137344E+01 exlk= 0.838257E+00 SSP= 0.839265E+00 SEDavg= 0.150638E+01 N= 14
3SED= 0.152526E+01 Area = 0.144650E+01 exlk= 0.838257E+00 SSP= 0.839265E+00 SEDavg= 0.150638E+01 N= 14
4SED= 0.147160E+01 Area = 0.245067E+01 exlk= 0.838257E+00 SSP= 0.839265E+00 SEDavg= 0.150638E+01 N= 14

1SED= 0.151031E+01 Area = 0.181628E+01 exlk= 0.838386E+00 SSP= 0.838256E+00 SEDavg= 0.150741E+01 N= 15
2SED= 0.149351E+01 Area = 0.138876E+01 exlk= 0.838386E+00 SSP= 0.838256E+00 SEDavg= 0.150741E+01 N= 15
3SED= 0.149459E+01 Area = 0.146141E+01 exlk= 0.838386E+00 SSP= 0.838256E+00 SEDavg= 0.150741E+01 N= 15
4SED= 0.153124E+01 Area = 0.240271E+01 exlk= 0.838386E+00 SSP= 0.838256E+00 SEDavg= 0.150741E+01 N= 15

1SED= 0.150325E+01 Area = 0.181904E+01 exlk= 0.838045E+00 SSP= 0.838385E+00 SEDavg= 0.150529E+01 N= 16
2SED= 0.151466E+01 Area = 0.137789E+01 exlk= 0.838045E+00 SSP= 0.838385E+00 SEDavg= 0.150529E+01 N= 16
3SED= 0.151395E+01 Area = 0.145086E+01 exlk= 0.838045E+00 SSP= 0.838385E+00 SEDavg= 0.150529E+01 N= 16
4SED= 0.148929E+01 Area = 0.243430E+01 exlk= 0.838045E+00 SSP= 0.838385E+00 SEDavg= 0.150529E+01 N= 16

1SED= 0.150878E+01 Area = 0.181691E+01 exlk= 0.838159E+00 SSP= 0.838045E+00 SEDavg= 0.150743E+01 N= 17
2SED= 0.150107E+01 Area = 0.138502E+01 exlk= 0.838159E+00 SSP= 0.838045E+00 SEDavg= 0.150743E+01 N= 17
3SED= 0.150155E+01 Area = 0.145780E+01 exlk= 0.838159E+00 SSP= 0.838045E+00 SEDavg= 0.150743E+01 N= 17
4SED= 0.151832E+01 Area = 0.241250E+01 exlk= 0.838159E+00 SSP= 0.838045E+00 SEDavg= 0.150743E+01 N= 17

1SED= 0.150455E+01 Area = 0.181824E+01 exlk= 0.838028E+00 SSP= 0.838158E+00 SEDavg= 0.150548E+01 N= 18
2SED= 0.150978E+01 Area = 0.138009E+01 exlk= 0.838028E+00 SSP= 0.838158E+00 SEDavg= 0.150548E+01 N= 18
3SED= 0.150946E+01 Area = 0.145301E+01 exlk= 0.838028E+00 SSP= 0.838158E+00 SEDavg= 0.150548E+01 N= 18
4SED= 0.149814E+01 Area = 0.242706E+01 exlk= 0.838028E+00 SSP= 0.838158E+00 SEDavg= 0.150548E+01 N= 18

1SED= 0.150628E+01 Area = 0.181728E+01 exlk= 0.838092E+00 SSP= 0.838028E+00 SEDavg= 0.150565E+01 N= 19
2SED= 0.150274E+01 Area = 0.138338E+01 exlk= 0.838092E+00 SSP= 0.838028E+00 SEDavg= 0.150565E+01 N= 19
3SED= 0.150296E+01 Area = 0.145621E+01 exlk= 0.838092E+00 SSP= 0.838028E+00 SEDavg= 0.150565E+01 N= 19
4SED= 0.151063E+01 Area = 0.241712E+01 exlk= 0.838092E+00 SSP= 0.838028E+00 SEDavg= 0.150565E+01 N= 19

1SED= 0.149165E+01 Area = 0.216836E+01 exlk= 0.100000E+01 SSP= 0.838028E+00 SEDavg= 0.149103E+01 N= 20
2SED= 0.148816E+01 Area = 0.165063E+01 exlk= 0.100000E+01 SSP= 0.838028E+00 SEDavg= 0.149103E+01 N= 20
3SED= 0.148837E+01 Area = 0.173753E+01 exlk= 0.100000E+01 SSP= 0.838028E+00 SEDavg= 0.149103E+01 N= 20
4SED= 0.149593E+01 Area = 0.288408E+01 exlk= 0.100000E+01 SSP= 0.838028E+00 SEDavg= 0.149103E+01 N= 20

fea3d2_out.txt

	istep	exlk	appliedforce	a(14)	a(15)	it	Weight	Area1	Area2	Area3	Area4	xlie 1	xlie 2	fin1
*	1	0.100000E+00	-0.20E+02	-.5597-316	0.3908-314	3	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13419624E+03	0.18746808E+03	-.22E+03
*	2	0.168218E+00	-0.34E+02	0.9180E-03	-.6381E-01	3	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13419527E+03	0.18746673E+03	-.38E+03
*	3	0.206585E+00	-0.41E+02	0.1523E-02	-.1067E+00	3	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13419471E+03	0.18746595E+03	-.47E+03
*	87	0.310484E+00	-0.62E+02	0.2785E-02	-.1981E+00	2	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13419314E+03	0.18746376E+03	-.72E+03
*	320	0.506051E+00	-0.10E+03	0.4808E-02	-.3512E+00	2	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13418991E+03	0.18745924E+03	-.12E+04
*	715	0.782859E+00	-0.16E+03	0.8372E-02	-.6451E+00	2	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13418420E+03	0.18745129E+03	-.22E+04
*	1376	0.999994E+00	-0.20E+03	0.1446E-01	-.1259E+01	1	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13417445E+03	0.18743767E+03	-.37E+04
*	1377	0.100000E+01	-0.20E+03	0.1447E-01	-.1260E+01	1	0.11506603E+03	0.216836E+01	0.165063E+01	0.173753E+01	0.288408E+01	0.13417444E+03	0.18743766E+03	-.37E+04

B.4 Optimum Khot 30 Element Star Example

Star dome 30 element truss

The input files:
Dome2_opt_mod_in.dat
36 3 19 1 1 1 4 0 0 0 1 12 1 1
30 0 0 0 0 0
6 12 24 4 4 4
1.e7 1.23 1.23 1.23 1.23 1.23 1.23 1.23 1.23 1.23 1.23
1 0. 0. 85.912

```
2  180.  -311.769  64.662
3  360.     0.    64.662
4  180.   311.769  64.662
5 -180.   311.769  64.662
6 -360.     0.    64.662
7 -180.  -311.769  64.662
8    0.  -623.538  21.709
9  360.  -623.538   0.
10 540.  -311.769  21.709
11  720.     0.    0.
12  540.  311.769  21.709
13  360.  623.538   0.
14    0.  623.538  21.709
15 -360.  623.538   0.
16 -540.  311.769  21.709
17 -720.     0.    0.
18 -540. -311.769  21.709
19 -360. -623.538   0.
  1  1  1      5
  2  1  1      4
  3  1  1      3
  4  1  1      2
  5  1  1      7
  6  1  1      6
  7  2  4      5
  8  2  3      4
  9  2  2      3
 10  2  7      2
 11  2  6      7
 12  2  5      6
 13  3  5     15
 14  4  5     14
 15  4  4     14
 16  3  4     13
 17  4  4     12
 18  4  3     12
 19  3  3     11
 20  4  3     10
 21  4  2     10
 22  3  2      9
 23  4  2      8
 24  4  7      8
 25  3  7     19
 26  4  7     18
 27  4  6     18
 28  3  6     17
 29  4  6     16
 30  4  5     16
 3   -2000.d0
 3 1  1.0
```

Dome2_opt_mod_in_areas.dat

```
 1.0d0
 1.0d0
 1.0d0
 1.0d0
```

dome2_sup.dat

```
 22   0.0
 23   0.0
 24   0.0
 25   0.0
 26   0.0
 27   0.0
 28   0.0
 29   0.0
 30   0.0
 31   0.0
 32   0.0
 33   0.0
```

```
 34   0.0
 35   0.0
 36   0.0
 37   0.0
 38   0.0
 39   0.0
 40   0.0
 41   0.0
 42   0.0
 43   0.0
 44   0.0
 45   0.0
 46   0.0
 47   0.0
 48   0.0
 49   0.0
 50   0.0
 51   0.0
 52   0.0
 53   0.0
 54   0.0
 55   0.0
 56   0.0
 57   0.0

            open(unit=5,file='dome2_opt_mod_in.dat',status='old')
            open(unit=8,file='dome2_opt_mod_in_areas.dat',status='old')

            open(unit=5,file='dome2_opt_mod_in.dat',status='old')

            open(unit=51,file='dome2_sup.dat',status='old')

            dSEDtol=1.d0
            Amin=.1d0

C** 3D Truss , Dome_in.dat & Dome2_in.dat
      itedes=1
       dl0 = .1d0
       dlmax = .1d0
        dlmin = .01d0
      exlmx = 1.d0
     toldis = 1.d-4
     tolfor = 1.d-4

c** "Dome2_opt_mod_in_initial.dat"**************************************************
       appliedforce = exlk * exfor(3)
       weightotal=0.d0
            Do 55 i=1,nel
            weightotal=weightotal+weight(i)
55          Continue

      if(istep .eq. 1) then
     write(15,53)
 53   format('  istep     exlk       appliedforce       a(2)
     $  a(3)       it    Weight     Area1        Area2
     $  Area3        Area4      xlie 1        xlie 2        fin1')
            endif

     write(15,50) '*',istep,exlk,appliedforce,
    c   dqtot(2),dqtot(3),it,weightotal,
    c   propsect(1),propsect(2),propsect(3),propsect(4),
    c   xlie(1),xlie(2),fin1(1)

50    format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
    c 1x,i5,1x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
    c  e12.6,2X,e12.6,1x,e8.2)
```

MSC-Nastran run deck
Khot_dome_2009.dat
CEND

```
 ECHO = NONE
 DISPLACEMENT = ALL
 OLOAD = ALL
 SPCFORCE = ALL
 FORCE = ALL
 STRESS = ALL
 SPC  = 1
 LOAD = 1
 NLPARM = 1
BEGIN BULK
$ ******************************************************************************
$  Written by : MSC/NASTRAN for Windows
$  Version    : 5.00
$  Translator : MSC/NASTRAN
$  From Model : E:\Mscn4w30\data\PhD\Optimization\Khot_2ele_GAH_symmetric.MOD
$  Date       : Sun Feb 02 18:02:07 2003
$  Output To  : E:\Mscn4w30\data\PhD\Optimization\Dome\Khot_000
$ ******************************************************************************
$
NLPARM       1     200          AUTO       5     200     UP     ALL+
+          0.01    0.01   1.E-7     30      25      4     0.2    0.5+
+             5                     20.            20.
NLPCI        1   MRIKS    0.25      4.                  12     900
PARAM,LGDISP,1
PARAM,PRGPST,NO
PARAM,AUTOSPC,YES
PARAM,K6ROT,100.
PARAM,MAXRATIO,1.E+8
PARAM,GRDPNT,0
CORD2C       1       0      0.      0.      0.      0.      0.      1.+MSC/NC1
+MSC/NC1     1.      0.     1.
CORD2S       2       0      0.      0.      0.      0.      0.      1.+MSC/NC2
+MSC/NC2     1.      0.     1.
$ MSC/NASTRAN for Windows Load Set 1 : Untitled
FORCE        1       2      0       1.      0.   -200.      0.
$ MSC/NASTRAN for Windows Constraint Set 1 : Untitled
SPC          1       1  12345       0.
SPC          1       2    345       0.
SPC          1       3  12345       0.
$ MSC/NASTRAN for Windows Property 1 : Untitled
PROD         1       1 6.4999       0.      0.      0.
$ MSC/NASTRAN for Windows Property 2 : Untitled
PROD         2       1 6.4999       0.      0.      0.
$ MSC/NASTRAN for Windows Material 1 : Untitled
MAT1         1   1.E+7                  0.      0.      0.
GRID         1       0      0.      0.      0.      0
GRID         2       0    125.      2.5     0.      0
GRID         3       0    250.      0.      0.      0
CROD         1       1      1       2
CROD         2       2      2       3
ENDDATA
```

SED_out.txt

```
 1SED= 0.194286E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 2SED= 0.194286E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 3SED= 0.194286E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 4SED= 0.194286E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 5SED= 0.194286E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 6SED= 0.194286E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 7SED= 0.126056E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 8SED= 0.126056E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
 9SED= 0.126056E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
10SED= 0.126056E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
11SED= 0.126056E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
12SED= 0.126056E+02 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
13SED= 0.740568E+00 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
14SED= 0.183099E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
15SED= 0.183099E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
16SED= 0.740568E+00 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
17SED= 0.183208E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
18SED= 0.183349E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
19SED= 0.740566E+00 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
20SED= 0.183349E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
21SED= 0.183208E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
```

22SED= 0.740568E+00 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
23SED= 0.183099E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
24SED= 0.183099E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
25SED= 0.740568E+00 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
26SED= 0.183208E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
27SED= 0.183349E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
28SED= 0.740566E+00 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
29SED= 0.183349E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1
30SED= 0.183208E-05 Area = 0.100000E+01 exlk= 0.654306E+00 SSP= 0.000000E+00 SEDavg= 0.655494E+01 N= 1

1SED= 0.143351E+02 Area = 0.111806E+01 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
2SED= 0.143351E+02 Area = 0.111806E+01 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
3SED= 0.143351E+02 Area = 0.111806E+01 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
4SED= 0.143351E+02 Area = 0.111806E+01 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
5SED= 0.143351E+02 Area = 0.111806E+01 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
6SED= 0.143351E+02 Area = 0.111806E+01 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
7SED= 0.169094E+02 Area = 0.842434E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
8SED= 0.169094E+02 Area = 0.842434E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
9SED= 0.169094E+02 Area = 0.842434E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
10SED= 0.169094E+02 Area = 0.842434E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
11SED= 0.169094E+02 Area = 0.842434E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
12SED= 0.169094E+02 Area = 0.842434E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
13SED= 0.176374E+02 Area = 0.131853E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
14SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
15SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
16SED= 0.176374E+02 Area = 0.131853E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
17SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
18SED= 0.471913E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
19SED= 0.176374E+02 Area = 0.131853E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
20SED= 0.471913E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
21SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
22SED= 0.176374E+02 Area = 0.131853E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
23SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
24SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
25SED= 0.176374E+02 Area = 0.131853E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
26SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
27SED= 0.471913E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
28SED= 0.176374E+02 Area = 0.131853E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
29SED= 0.471913E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2
30SED= 0.471912E+01 Area = 0.100000E+00 exlk= 0.632868E+00 SSP= 0.654304E+00 SEDavg= 0.116640E+02 N= 2

1SED= 0.160263E+02 Area = 0.110883E+01 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
2SED= 0.160263E+02 Area = 0.110883E+01 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
3SED= 0.160263E+02 Area = 0.110883E+01 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
4SED= 0.160263E+02 Area = 0.110883E+01 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
5SED= 0.160263E+02 Area = 0.110883E+01 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
6SED= 0.160263E+02 Area = 0.110883E+01 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
7SED= 0.153015E+02 Area = 0.927540E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
8SED= 0.153015E+02 Area = 0.927540E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
9SED= 0.153015E+02 Area = 0.927540E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
10SED= 0.153015E+02 Area = 0.927540E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
11SED= 0.153015E+02 Area = 0.927540E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
12SED= 0.153015E+02 Area = 0.927540E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
13SED= 0.163549E+02 Area = 0.149098E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
14SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
15SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
16SED= 0.163549E+02 Area = 0.149098E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
17SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
18SED= 0.440872E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
19SED= 0.163549E+02 Area = 0.149098E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
20SED= 0.440872E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
21SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
22SED= 0.163549E+02 Area = 0.149098E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
23SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
24SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
25SED= 0.163549E+02 Area = 0.149098E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
26SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
27SED= 0.440872E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
28SED= 0.163549E+02 Area = 0.149098E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
29SED= 0.440872E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3
30SED= 0.440871E+01 Area = 0.100000E+00 exlk= 0.662298E+00 SSP= 0.632868E+00 SEDavg= 0.113000E+02 N= 3

1SED= 0.155456E+02 Area = 0.120629E+01 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
2SED= 0.155456E+02 Area = 0.120629E+01 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
3SED= 0.155456E+02 Area = 0.120629E+01 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
4SED= 0.155456E+02 Area = 0.120629E+01 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
5SED= 0.155456E+02 Area = 0.120629E+01 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
6SED= 0.155456E+02 Area = 0.120629E+01 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
7SED= 0.157382E+02 Area = 0.978606E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
8SED= 0.157382E+02 Area = 0.978606E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
9SED= 0.157382E+02 Area = 0.978606E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
10SED= 0.157382E+02 Area = 0.978606E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
11SED= 0.157382E+02 Area = 0.978606E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
12SED= 0.157382E+02 Area = 0.978606E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
13SED= 0.164459E+02 Area = 0.164398E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
14SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
15SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
16SED= 0.164459E+02 Area = 0.164398E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
17SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
18SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
19SED= 0.164459E+02 Area = 0.164398E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
20SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
21SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4

22SED= 0.164459E+02 Area = 0.164398E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
23SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
24SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
25SED= 0.164459E+02 Area = 0.164398E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
26SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
27SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
28SED= 0.164459E+02 Area = 0.164398E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
29SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4
30SED= 0.440294E+01 Area = 0.100000E+00 exlk= 0.710032E+00 SSP= 0.662297E+00 SEDavg= 0.113071E+02 N= 4

1SED= 0.156819E+02 Area = 0.129120E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
2SED= 0.156819E+02 Area = 0.129120E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
3SED= 0.156819E+02 Area = 0.129120E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
4SED= 0.156819E+02 Area = 0.129120E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
5SED= 0.156819E+02 Area = 0.129120E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
6SED= 0.156819E+02 Area = 0.129120E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
7SED= 0.155558E+02 Area = 0.105668E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
8SED= 0.155558E+02 Area = 0.105668E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
9SED= 0.155558E+02 Area = 0.105668E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
10SED= 0.155558E+02 Area = 0.105668E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
11SED= 0.155558E+02 Area = 0.105668E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
12SED= 0.155558E+02 Area = 0.105668E+01 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
13SED= 0.163077E+02 Area = 0.183145E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
14SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
15SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
16SED= 0.163077E+02 Area = 0.183145E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
17SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
18SED= 0.437025E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
19SED= 0.163077E+02 Area = 0.183145E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
20SED= 0.437025E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
21SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
22SED= 0.163077E+02 Area = 0.183145E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
23SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
24SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
25SED= 0.163077E+02 Area = 0.183145E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
26SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
27SED= 0.437025E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
28SED= 0.163077E+02 Area = 0.183145E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
29SED= 0.437025E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5
30SED= 0.437024E+01 Area = 0.100000E+00 exlk= 0.763873E+00 SSP= 0.710030E+00 SEDavg= 0.112572E+02 N= 5

1SED= 0.155909E+02 Area = 0.140259E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
2SED= 0.155909E+02 Area = 0.140259E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
3SED= 0.155909E+02 Area = 0.140259E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
4SED= 0.155909E+02 Area = 0.140259E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
5SED= 0.155909E+02 Area = 0.140259E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
6SED= 0.155909E+02 Area = 0.140259E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
7SED= 0.156135E+02 Area = 0.114079E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
8SED= 0.156135E+02 Area = 0.114079E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
9SED= 0.156135E+02 Area = 0.114079E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
10SED= 0.156135E+02 Area = 0.114079E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
11SED= 0.156135E+02 Area = 0.114079E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
12SED= 0.156135E+02 Area = 0.114079E+01 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
13SED= 0.162303E+02 Area = 0.204982E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
14SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
15SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
16SED= 0.162303E+02 Area = 0.204982E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
17SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
18SED= 0.433827E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
19SED= 0.162303E+02 Area = 0.204982E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
20SED= 0.433827E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
21SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
22SED= 0.162303E+02 Area = 0.204982E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
23SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
24SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
25SED= 0.162303E+02 Area = 0.204982E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
26SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
27SED= 0.433827E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
28SED= 0.162303E+02 Area = 0.204982E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
29SED= 0.433827E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6
30SED= 0.433826E+01 Area = 0.100000E+00 exlk= 0.827755E+00 SSP= 0.763872E+00 SEDavg= 0.112222E+02 N= 6

1SED= 0.156252E+02 Area = 0.152947E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
2SED= 0.156252E+02 Area = 0.152947E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
3SED= 0.156252E+02 Area = 0.152947E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
4SED= 0.156252E+02 Area = 0.152947E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
5SED= 0.156252E+02 Area = 0.152947E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
6SED= 0.156252E+02 Area = 0.152947E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
7SED= 0.155686E+02 Area = 0.124548E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
8SED= 0.155686E+02 Area = 0.124548E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
9SED= 0.155686E+02 Area = 0.124548E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
10SED= 0.155686E+02 Area = 0.124548E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
11SED= 0.155686E+02 Area = 0.124548E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
12SED= 0.155686E+02 Area = 0.124548E+01 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
13SED= 0.161334E+02 Area = 0.231088E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
14SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
15SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
16SED= 0.161334E+02 Area = 0.231088E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
17SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
18SED= 0.430826E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
19SED= 0.161334E+02 Area = 0.231088E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
20SED= 0.430826E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
21SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7

```
22SED= 0.161334E+02 Area = 0.231088E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
23SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
24SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
25SED= 0.161334E+02 Area = 0.231088E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
26SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
27SED= 0.430826E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
28SED= 0.161334E+02 Area = 0.231088E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
29SED= 0.430826E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
30SED= 0.430825E+01 Area = 0.100000E+00 exlk= 0.903754E+00 SSP= 0.827755E+00 SEDavg= 0.111887E+02 N= 7
```

```
1SED= 0.156044E+02 Area = 0.168767E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
2SED= 0.156044E+02 Area = 0.168767E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
3SED= 0.156044E+02 Area = 0.168767E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
4SED= 0.156044E+02 Area = 0.168767E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
5SED= 0.156044E+02 Area = 0.168767E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
6SED= 0.156044E+02 Area = 0.168767E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
7SED= 0.156121E+02 Area = 0.136980E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
8SED= 0.156121E+02 Area = 0.136980E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
9SED= 0.156121E+02 Area = 0.136980E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
10SED= 0.156121E+02 Area = 0.136980E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
11SED= 0.156121E+02 Area = 0.136980E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
12SED= 0.156121E+02 Area = 0.136980E+01 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
13SED= 0.160748E+02 Area = 0.262472E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
14SED= 0.428412E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
15SED= 0.428412E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
16SED= 0.160748E+02 Area = 0.262472E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
17SED= 0.428411E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
18SED= 0.428413E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
19SED= 0.160748E+02 Area = 0.262472E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
20SED= 0.428413E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
21SED= 0.428411E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
22SED= 0.160748E+02 Area = 0.262472E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
23SED= 0.428412E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
24SED= 0.428412E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
25SED= 0.160748E+02 Area = 0.262472E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
26SED= 0.428411E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
27SED= 0.428413E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
28SED= 0.160748E+02 Area = 0.262472E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
29SED= 0.428413E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
30SED= 0.428411E+01 Area = 0.100000E+00 exlk= 0.996220E+00 SSP= 0.903754E+00 SEDavg= 0.111719E+02 N= 8
```

---

fea3d2_out.txt

istep	exlk	appliedforce	a(2)	a(3)	it	Weight	Area1	Area2	Area3	Area4	xlie 1	xlie 2	fin1	
* 1	0.100000E+00	-0.20E+03	0.1932-314	0.2112-313	3	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360614E+03	0.360614E+03	-.58E+03	
* 2	0.123792E+00	-0.25E+03	0.1709E-17	-.4757E+00	3	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360611E+03	0.360611E+03	-.72E+03	
* 3	0.136007E+00	-0.27E+03	0.4762E-17	-.5763E+00	3	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360610E+03	0.360610E+03	-.79E+03	
* 127	0.400938E+00	-0.80E+03	0.3300E-16	-.1956E+01	2	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360574E+03	0.360574E+03	-.25E+04	
* 384	0.771478E+00	-0.15E+04	-.7842E-17	-.4500E+01	2	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360510E+03	0.360510E+03	-.54E+04	
* 527	0.901404E+00	-0.18E+04	0.2709E-16	-.5919E+01	2	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360478E+03	0.360478E+03	-.69E+04	
* 704	0.985003E+00	-0.20E+04	-.3483E-16	-.7660E+01	1	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360442E+03	0.360442E+03	-.86E+04	
* 801	0.996220E+00	-0.20E+04	0.3495E-16	-.8598E+01	1	0.762154E+03	0.168767E+01	0.136980E+01	0.262472E+00	0.100000E+00	0.360425E+03	0.360425E+03	-.94E+04	

B.5 Optimum Khot Large Shallow Truss Example

Shallow truss half-symmetry model with two point loads

The input files:

shallow_truss_in.dat

```
16  3  13   2  1   1  23  0 0 0 1 12 1 1
23 0 0 0 0 0
 6  12 24  4  4  4
1.e7  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23  1.23
 1  -10.94   4.94    0.
 2    0.     0.      0.
 3   -1.19  11.94    0.
 4   28.66   2.87    0.
 5   56.14  17.67    0.
 6   85.99   8.59    0.
 7   113.5  23.41    0.
 8   143.3  14.33    0.
 9   170.8  29.14    0.
10   200.7  20.07    0.
```

```
11   228.1  34.87    0.
12   229.3  22.93    0.
13   239.1  29.94    0.
 1     1     1     3
 2     2     1     2
 3     3     2     3
 4     4     3     4
 5     5     3     5
 6     6     2     4
 7     7     4     5
 8     8     5     6
 9     9     5     7
10    10     4     6
11    11     6     7
12    12     7     8
13    13     7     9
14    14     6     8
15    15     8     9
16    16     9    10
17    17     9    11
18    18     8    10
19    19    10    11
20    20    10    12
21    21    11    12
22    22    12    13
23    23    11    13
  20 -300.0
  38 -600.0
  38  1  1.0
```

shallow_truss_areas.dat

```
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
 2.0
```

shallow_truss_sup.dat

```
 1  0.0
 2  0.0
 3  0.0
 6  0.0
 9  0.0
12  0.0
15  0.0
18  0.0
21  0.0
24  0.0
27  0.0
30  0.0
33  0.0
36  0.0
```

```
37  0.0
39  0.0
```

fearc_opt3_shallow_truss_2009.for

```
            open(unit=5,file='shallow_truss_in.dat',status='old')
            open(unit=8,file='shallow_truss_areas.dat',status='old')

            open(4,file='SED_out.txt',status='unknown')

            open(unit=5,file='shallow_truss_in.dat',status='old')

            open(unit=51,file='shallow_truss_sup.dat',status='old')

            dSEDtol=1.d-1 !shallow truss
            Amin=.1d0

             Subroutine rikarc
C** Shallow_truss & Khot 4element
        itedes=1
        dl0 = .1d0
        dlmax = .10d0
        dlmin = .001d0
        exlmx = 1.d0
        toldis = 1.d-4
        tolfor = 1.d-4

    subroutine result
c**"shallow_truss_initial.dat"************************************************
      appliedforce = exlk * exfor(38)
      weightotal=0.d0
            Do 55 i=1,nel
            weightotal=weightotal+weight(i)
55          Continue

     if(istep .eq. 1) then
    write(15,53)
 53  format('  istep     exlk       appliedforce       a(28)
   $ a(38)       it    Weight     Area1        Area2
   $ Area3        Area4      xlie 10        xlie 15        fin1')
            endif

    write(15,50) '*',istep,exlk,appliedforce,
   c   dqtot(26),dqtot(29),it,weightotal,
   c   propsect(1),propsect(2),propsect(3),propsect(4),
   c   xlie(16),xlie(20),fin1(20)

50   format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
   c 1x,i5,1x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
   c  e12.6,2X,e12.6,1x,e8.2)
```

MSC-Nastran run deck

shallow_Khot_truss_msc_2009.NAS

```
ID E:\Mscn4w30\data\PhD\Opti,FEMAP
SOL NLSTATIC
TIME 10000
CEND
 ECHO = NONE
 DISPLACEMENT = ALL
 SPC = 1
 LOAD = 1
 NLPARM = 1
BEGIN BULK
$ ******************************************************************************
$  Written by : FEMAP
$  Version    : 5.00
$  Translator : MSC/NASTRAN
$  From Model : E:\Mscn4w30\data\PhD\Optimization\shallow_Khot_final.MOD
$  Date       : Sun Feb 02 18:27:29 2003
$ ******************************************************************************
$
```

```
NLPARM       1    900         ITER     1    900     PW    YES+
+         0.001  0.001  I.E-7     3     25     4    0.2    0.5+
+           5                   20.          20.
NLPC1        1  MRIKS   0.25     4.               500    900
PARAM,LGDISP,1
PARAM,AUTOSPC,YES
PARAM,GRDPNT,0
CORD2C       1     0     0.     0.     0.     0.     0.     1.+FEMAPC1
+FEMAPC1     1.    0.    1.
CORD2S       2     0     0.     0.     0.     0.     0.     1.+FEMAPC2
+FEMAPC2     1.    0.    1.
$ FEMAP Load Set 1 : Untitled
FORCE        1     13     0     1.     0.   -600.     0.
FORCE        1     7      0     1.     0.   -300.     0.
$ FEMAP Constraint Set 1 : Untitled
SPC          1     1   12345     0.
SPC          1     2    345      0.
SPC          1     3    345      0.
SPC          1     4    345      0.
SPC          1     5    345      0.
SPC          1     6    345      0.
SPC          1     7    345      0.
SPC          1     8    345      0.
SPC          1     9    345      0.
SPC          1    10    345      0.
SPC          1    11    345      0.
SPC          1    12    345      0.
SPC          1    13   1345      0.
$ FEMAP Property 1 : Untitled
PROD         1     1  2.4747     0.     0.     0.
$ FEMAP Property 2 : Untitled
PROD         2     1  2.3695     0.     0.     0.
$ FEMAP Property 3 : Untitled
PROD         3     1  1.1837     0.     0.     0.
$ FEMAP Property 4 : Untitled
PROD         4     1  0.2609     0.     0.     0.
$ FEMAP Property 5 : Untitled
PROD         5     1  2.3694     0.     0.     0.
$ FEMAP Property 6 : Untitled
PROD         6     1  2.0551     0.     0.     0.
$ FEMAP Property 7 : Untitled
PROD         7     1  0.2259     0.     0.     0.
$ FEMAP Property 8 : Untitled
PROD         8     1  0.2227     0.     0.     0.
$ FEMAP Property 9 : Untitled
PROD         9     1  2.7512     0.     0.     0.
$ FEMAP Property 10 : Untitled
PROD        10     1  1.6353     0.     0.     0.
$ FEMAP Property 11 : Untitled
PROD        11     1  0.1945     0.     0.     0.
$ FEMAP Property 12 : Untitled
PROD        12     1  0.1941     0.     0.     0.
$ FEMAP Property 13 : Untitled
PROD        13     1  2.7463     0.     0.     0.
$ FEMAP Property 14 : Untitled
PROD        14     1  1.2751     0.     0.     0.
$ FEMAP Property 15 : Untitled
PROD        15     1  0.223      0.     0.     0.
$ FEMAP Property 16 : Untitled
PROD        16     1  0.2249     0.     0.     0.
$ FEMAP Property 17 : Untitled
PROD        17     1  2.3649     0.     0.     0.
$ FEMAP Property 18 : Untitled
PROD        18     1  1.6351     0.     0.     0.
$ FEMAP Property 19 : Untitled
PROD        19     1  0.259      0.     0.     0.
$ FEMAP Property 20 : Untitled
PROD        20     1  0.0524     0.     0.     0.
$ FEMAP Property 21 : Untitled
PROD        21     1  1.1756     0.     0.     0.
```

```
$ FEMAP Property 22 : Untitled
PROD       22       1 2.3626     0.     0.     0.
$ FEMAP Property 23 : Untitled
PROD       23       1 2.4687     0.     0.     0.
$ FEMAP Material 1 : Untitled
MAT1        1   1.E+7          0.1    0.     0.     0.
GRID        1       0 -10.94    4.94     0.       0
GRID        2       0     0.     0.     0.       0
GRID        3       0 -1.19    11.94     0.       0
GRID        4       0  28.66    2.87     0.       0
GRID        5       0  56.14   17.67     0.       0
GRID        6       0  85.99    8.59     0.       0
GRID        7       0  113.5   23.41     0.       0
GRID        8       0  143.3   14.33     0.       0
GRID        9       0  170.8   29.14     0.       0
GRID       10       0  200.7   20.07     0.       0
GRID       11       0  228.1   34.87     0.       0
GRID       12       0  229.3   22.93     0.       0
GRID       13       0  239.1   29.94     0.       0
CROD        1       1       1       3
CROD        2       2       1       2
CROD        3       3       2       3
CROD        4       4       3       4
CROD        5       5       3       5
CROD        6       6       2       4
CROD        7       7       4       5
CROD        8       8       5       6
CROD        9       9       5       7
CROD       10      10       4       6
CROD       11      11       6       7
CROD       12      12       7       8
CROD       13      13       7       9
CROD       14      14       6       8
CROD       15      15       8       9
CROD       16      16       9      10
CROD       17      17       9      11
CROD       18      18       8      10
CROD       19      19      10      11
CROD       20      20      10      12
CROD       21      21      11      12
CROD       22      22      12      13
CROD       23      23      11      13
ENDDATA
```

SED_out.txt

```
 1SED= 0.265608E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 2SED= 0.230073E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 3SED= 0.574825E+00 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 4SED= 0.209752E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 5SED= 0.238759E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 6SED= 0.172523E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 7SED= 0.207116E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 8SED= 0.199578E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
 9SED= 0.327102E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
10SED= 0.109587E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
11SED= 0.193169E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
12SED= 0.192245E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
13SED= 0.323500E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
14SED= 0.621099E+00 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
15SED= 0.196568E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
16SED= 0.201948E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
17SED= 0.236487E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
18SED= 0.109313E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
19SED= 0.206514E-01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
20SED= 0.171458E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
21SED= 0.564183E+00 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
22SED= 0.227937E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
23SED= 0.262691E+01 Area = 0.200000E+01 exlk= 0.100006E+01 SSP= 0.000000E+00 SEDavg= 0.124657E+01 N= 1
-------------------------------------------------------------------------------------------------------------
 1SED= 0.258574E+01 Area = 0.202805E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
 2SED= 0.230773E+01 Area = 0.199913E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
 3SED= 0.760755E+00 Area = 0.174024E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
 4SED= 0.531954E-01 Area = 0.124976E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
 5SED= 0.237221E+01 Area = 0.200655E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
 6SED= 0.183304E+01 Area = 0.194240E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
 7SED= 0.527386E-01 Area = 0.124818E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
```

8SED= 0.514468E-01 Area = 0.124356E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
9SED= 0.304891E+01 Area = 0.207073E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
10SED= 0.127863E+01 Area = 0.185622E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
11SED= 0.502497E-01 Area = 0.123951E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
12SED= 0.500537E-01 Area = 0.123891E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
13SED= 0.302202E+01 Area = 0.206844E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
14SED= 0.813732E+00 Area = 0.175376E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
15SED= 0.508197E-01 Area = 0.124167E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
16SED= 0.516786E-01 Area = 0.124503E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
17SED= 0.235416E+01 Area = 0.200463E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
18SED= 0.127605E+01 Area = 0.185576E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
19SED= 0.525394E-01 Area = 0.124781E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
20SED= 0.182400E+01 Area = 0.194120E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
21SED= 0.749470E+00 Area = 0.173699E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
22SED= 0.229058E+01 Area = 0.199727E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2
23SED= 0.256301E+01 Area = 0.202581E+01 exlk= 0.100047E+01 SSP= 0.100000E+00 SEDavg= 0.128229E+01 N= 2

1SED= 0.250431E+01 Area = 0.206008E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
2SED= 0.228870E+01 Area = 0.200774E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
3SED= 0.941807E+00 Area = 0.156416E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
4SED= 0.111123E+00 Area = 0.860914E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
5SED= 0.233582E+01 Area = 0.202075E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
6SED= 0.190363E+01 Area = 0.190635E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
7SED= 0.110519E+00 Area = 0.859085E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
8SED= 0.108798E+00 Area = 0.853786E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
9SED= 0.285349E+01 Area = 0.213838E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
10SED= 0.143008E+01 Area = 0.175733E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
11SED= 0.106969E+00 Area = 0.849002E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
12SED= 0.106629E+00 Area = 0.848263E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
13SED= 0.283335E+01 Area = 0.213412E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
14SED= 0.997726E+00 Area = 0.158696E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
15SED= 0.107727E+00 Area = 0.851444E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
16SED= 0.108732E+00 Area = 0.855178E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
17SED= 0.232161E+01 Area = 0.201728E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
18SED= 0.142776E+01 Area = 0.175653E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
19SED= 0.110029E+00 Area = 0.858510E+00 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
20SED= 0.189612E+01 Area = 0.190423E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
21SED= 0.930616E+00 Area = 0.155890E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
22SED= 0.227510E+01 Area = 0.200437E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3
23SED= 0.248669E+01 Area = 0.205599E+01 exlk= 0.100020E+01 SSP= 0.100000E+00 SEDavg= 0.131728E+01 N= 3

1SED= 0.242158E+01 Area = 0.209534E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
2SED= 0.225511E+01 Area = 0.202380E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
3SED= 0.110812E+01 Area = 0.144270E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
4SED= 0.199219E+00 Area = 0.641270E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
5SED= 0.228922E+01 Area = 0.204107E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
6SED= 0.194610E+01 Area = 0.188653E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
7SED= 0.198616E+00 Area = 0.639559E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
8SED= 0.196814E+00 Area = 0.634618E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
9SED= 0.268594E+01 Area = 0.220355E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
10SED= 0.155034E+01 Area = 0.169001E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
11SED= 0.194412E+00 Area = 0.629993E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
12SED= 0.193906E+00 Area = 0.629244E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
13SED= 0.267076E+01 Area = 0.219761E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
14SED= 0.116334E+01 Area = 0.147220E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
15SED= 0.195253E+00 Area = 0.632252E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
16SED= 0.196036E+00 Area = 0.635614E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
17SED= 0.227808E+01 Area = 0.203632E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
18SED= 0.154831E+01 Area = 0.168897E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
19SED= 0.197653E+00 Area = 0.638847E+00 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
20SED= 0.193996E+01 Area = 0.188368E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
21SED= 0.109758E+01 Area = 0.143614E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
22SED= 0.224438E+01 Area = 0.201920E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4
23SED= 0.240795E+01 Area = 0.208970E+01 exlk= 0.100044E+01 SSP= 0.100000E+00 SEDavg= 0.135560E+01 N= 4

1SED= 0.233327E+01 Area = 0.213358E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
2SED= 0.220546E+01 Area = 0.204611E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
3SED= 0.124900E+01 Area = 0.135857E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
4SED= 0.315147E+00 Area = 0.508651E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
5SED= 0.222995E+01 Area = 0.206668E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
6SED= 0.196020E+01 Area = 0.187943E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
7SED= 0.314772E+00 Area = 0.507140E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
8SED= 0.313372E+00 Area = 0.502763E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
9SED= 0.253376E+01 Area = 0.226714E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
10SED= 0.163579E+01 Area = 0.164580E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
11SED= 0.310565E+00 Area = 0.498487E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
12SED= 0.309902E+00 Area = 0.497764E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
13SED= 0.252229E+01 Area = 0.225974E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
14SED= 0.130041E+01 Area = 0.139311E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
15SED= 0.311372E+00 Area = 0.500490E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
16SED= 0.311491E+00 Area = 0.503353E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
17SED= 0.222127E+01 Area = 0.206086E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
18SED= 0.163406E+01 Area = 0.164457E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
19SED= 0.313171E+00 Area = 0.506329E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
20SED= 0.195525E+01 Area = 0.187600E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
21SED= 0.123949E+01 Area = 0.135110E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
22SED= 0.219706E+01 Area = 0.204049E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5
23SED= 0.232275E+01 Area = 0.212664E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.139304E+01 N= 5

1SED= 0.224623E+01 Area = 0.217459E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
2SED= 0.214834E+01 Area = 0.207372E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
3SED= 0.136298E+01 Area = 0.130079E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
4SED= 0.451738E+00 Area = 0.424367E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6

5SED= 0.216587E+01 Area = 0.209688E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
6SED= 0.195500E+01 Area = 0.188246E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
7SED= 0.451803E+00 Area = 0.423056E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
8SED= 0.451230E+00 Area = 0.419218E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
9SED= 0.239889E+01 Area = 0.232984E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
10SED= 0.169255E+01 Area = 0.161890E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
11SED= 0.448211E+00 Area = 0.415279E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
12SED= 0.447424E+00 Area = 0.414588E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
13SED= 0.239020E+01 Area = 0.232118E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
14SED= 0.140862E+01 Area = 0.133925E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
15SED= 0.448916E+00 Area = 0.417056E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
16SED= 0.448032E+00 Area = 0.419458E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
17SED= 0.215912E+01 Area = 0.209016E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
18SED= 0.169110E+01 Area = 0.161753E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
19SED= 0.449480E+00 Area = 0.422165E+00 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
20SED= 0.195105E+01 Area = 0.187855E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
21SED= 0.135466E+01 Area = 0.129265E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
22SED= 0.214180E+01 Area = 0.206723E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6
23SED= 0.223812E+01 Area = 0.216654E+01 exlk= 0.100029E+01 SSP= 0.100000E+00 SEDavg= 0.143049E+01 N= 6

1SED= 0.215867E+01 Area = 0.221772E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
2SED= 0.208391E+01 Area = 0.210545E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
3SED= 0.144788E+01 Area = 0.126195E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
4SED= 0.597522E+00 Area = 0.368651E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
5SED= 0.209643E+01 Area = 0.213069E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
6SED= 0.193249E+01 Area = 0.189333E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
7SED= 0.598181E+00 Area = 0.367517E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
8SED= 0.598680E+00 Area = 0.364137E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
9SED= 0.227521E+01 Area = 0.239172E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
10SED= 0.172254E+01 Area = 0.160494E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
11SED= 0.595636E+00 Area = 0.360473E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
12SED= 0.594772E+00 Area = 0.359810E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
13SED= 0.226862E+01 Area = 0.238198E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
14SED= 0.148677E+01 Area = 0.130355E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
15SED= 0.596215E+00 Area = 0.362072E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
16SED= 0.594187E+00 Area = 0.364086E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
17SED= 0.209120E+01 Area = 0.212320E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
18SED= 0.172135E+01 Area = 0.160344E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
19SED= 0.595142E+00 Area = 0.366553E+00 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
20SED= 0.192936E+01 Area = 0.188901E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
21SED= 0.144080E+01 Area = 0.125328E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
22SED= 0.207883E+01 Area = 0.209822E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7
23SED= 0.215243E+01 Area = 0.220871E+01 exlk= 0.100031E+01 SSP= 0.100000E+00 SEDavg= 0.146334E+01 N= 7

1SED= 0.207437E+01 Area = 0.226184E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
2SED= 0.201733E+01 Area = 0.213978E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
3SED= 0.150735E+01 Area = 0.123667E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
4SED= 0.741864E+00 Area = 0.330664E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
5SED= 0.202627E+01 Area = 0.216673E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
6SED= 0.189919E+01 Area = 0.190974E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
7SED= 0.743164E+00 Area = 0.329684E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
8SED= 0.744757E+00 Area = 0.326679E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
9SED= 0.216366E+01 Area = 0.245217E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
10SED= 0.173251E+01 Area = 0.160034E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
11SED= 0.741830E+00 Area = 0.323227E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
12SED= 0.740938E+00 Area = 0.322586E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
13SED= 0.215864E+01 Area = 0.244146E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
14SED= 0.153942E+01 Area = 0.128082E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
15SED= 0.742296E+00 Area = 0.324692E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
16SED= 0.739199E+00 Area = 0.326387E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
17SED= 0.202223E+01 Area = 0.215858E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
18SED= 0.173154E+01 Area = 0.159873E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
19SED= 0.739509E+00 Area = 0.328652E+00 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
20SED= 0.189673E+01 Area = 0.190508E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
21SED= 0.150144E+01 Area = 0.122757E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
22SED= 0.201338E+01 Area = 0.213192E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8
23SED= 0.206957E+01 Area = 0.225200E+01 exlk= 0.100043E+01 SSP= 0.100000E+00 SEDavg= 0.149075E+01 N= 8

1SED= 0.199445E+01 Area = 0.230550E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
2SED= 0.195090E+01 Area = 0.217501E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
3SED= 0.154507E+01 Area = 0.122092E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
4SED= 0.875875E+00 Area = 0.304113E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
5SED= 0.195731E+01 Area = 0.220338E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
6SED= 0.185897E+01 Area = 0.192950E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
7SED= 0.877773E+00 Area = 0.303264E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
8SED= 0.880304E+00 Area = 0.300564E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
9SED= 0.206309E+01 Area = 0.251006E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
10SED= 0.172734E+01 Area = 0.160212E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
11SED= 0.877597E+00 Area = 0.297271E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
12SED= 0.876719E+00 Area = 0.296646E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
13SED= 0.205926E+01 Area = 0.249852E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
14SED= 0.157083E+01 Area = 0.126718E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
15SED= 0.877971E+00 Area = 0.298638E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
16SED= 0.874034E+00 Area = 0.300070E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
17SED= 0.195418E+01 Area = 0.219465E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
18SED= 0.172655E+01 Area = 0.160042E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
19SED= 0.873659E+00 Area = 0.302166E+00 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
20SED= 0.185703E+01 Area = 0.192455E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
21SED= 0.154022E+01 Area = 0.121147E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
22SED= 0.194785E+01 Area = 0.216659E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9
23SED= 0.199074E+01 Area = 0.229494E+01 exlk= 0.100021E+01 SSP= 0.100000E+00 SEDavg= 0.151121E+01 N= 9

1SED= 0.192441E+01 Area = 0.234725E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10

2SED= 0.189106E+01 Area = 0.220952E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
3SED= 0.156909E+01 Area = 0.121170E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
4SED= 0.996246E+00 Area = 0.285161E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
5SED= 0.189569E+01 Area = 0.223907E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
6SED= 0.181944E+01 Area = 0.195067E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
7SED= 0.998617E+00 Area = 0.284427E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
8SED= 0.100184E+01 Area = 0.281976E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
9SED= 0.197750E+01 Area = 0.256418E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
10SED= 0.171559E+01 Area = 0.160785E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
11SED= 0.999393E+00 Area = 0.278801E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
12SED= 0.998561E+00 Area = 0.278187E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
13SED= 0.197455E+01 Area = 0.255191E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
14SED= 0.158938E+01 Area = 0.125969E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
15SED= 0.999708E+00 Area = 0.280095E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
16SED= 0.995223E+00 Area = 0.281312E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
17SED= 0.189325E+01 Area = 0.222984E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
18SED= 0.171496E+01 Area = 0.160607E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
19SED= 0.994238E+00 Area = 0.283264E+00 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
20SED= 0.181791E+01 Area = 0.194546E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
21SED= 0.156514E+01 Area = 0.120194E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
22SED= 0.188868E+01 Area = 0.220062E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10
23SED= 0.192154E+01 Area = 0.233606E+01 exlk= 0.100052E+01 SSP= 0.100000E+00 SEDavg= 0.152791E+01 N= 10

1SED= 0.186159E+01 Area = 0.238593E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
2SED= 0.183599E+01 Area = 0.224200E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
3SED= 0.158125E+01 Area = 0.120678E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
4SED= 0.109941E+01 Area = 0.271391E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
5SED= 0.183934E+01 Area = 0.227254E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
6SED= 0.178015E+01 Area = 0.197172E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
7SED= 0.110211E+01 Area = 0.270756E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
8SED= 0.110575E+01 Area = 0.268509E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
9SED= 0.190283E+01 Area = 0.261353E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
10SED= 0.169832E+01 Area = 0.161568E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
11SED= 0.110360E+01 Area = 0.265421E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
12SED= 0.110283E+01 Area = 0.264814E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
13SED= 0.190056E+01 Area = 0.260064E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
14SED= 0.159697E+01 Area = 0.125619E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
15SED= 0.110387E+01 Area = 0.266661E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
16SED= 0.109913E+01 Area = 0.267700E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
17SED= 0.183745E+01 Area = 0.226289E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
18SED= 0.169781E+01 Area = 0.161383E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
19SED= 0.109765E+01 Area = 0.269531E+00 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
20SED= 0.177895E+01 Area = 0.196629E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
21SED= 0.157806E+01 Area = 0.119676E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
22SED= 0.183414E+01 Area = 0.223269E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11
23SED= 0.185936E+01 Area = 0.237420E+01 exlk= 0.100053E+01 SSP= 0.100000E+00 SEDavg= 0.153901E+01 N= 11

1SED= 0.180571E+01 Area = 0.242077E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
2SED= 0.178599E+01 Area = 0.227159E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
3SED= 0.158478E+01 Area = 0.120458E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
4SED= 0.118497E+01 Area = 0.261227E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
5SED= 0.178844E+01 Area = 0.230296E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
6SED= 0.174241E+01 Area = 0.199159E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
7SED= 0.118785E+01 Area = 0.260680E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
8SED= 0.119166E+01 Area = 0.258602E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
9SED= 0.183787E+01 Area = 0.265751E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
10SED= 0.167795E+01 Area = 0.162429E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
11SED= 0.118982E+01 Area = 0.255578E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
12SED= 0.118913E+01 Area = 0.254976E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
13SED= 0.183611E+01 Area = 0.264409E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
14SED= 0.159680E+01 Area = 0.125514E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
15SED= 0.119003E+01 Area = 0.256778E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
16SED= 0.118529E+01 Area = 0.257668E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
17SED= 0.178696E+01 Area = 0.229293E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
18SED= 0.167754E+01 Area = 0.162238E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
19SED= 0.118345E+01 Area = 0.259395E+00 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
20SED= 0.174147E+01 Area = 0.198596E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
21SED= 0.158222E+01 Area = 0.119433E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
22SED= 0.178455E+01 Area = 0.226193E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12
23SED= 0.180397E+01 Area = 0.240858E+01 exlk= 0.100007E+01 SSP= 0.100000E+00 SEDavg= 0.154500E+01 N= 12

1SED= 0.176091E+01 Area = 0.245142E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
2SED= 0.174566E+01 Area = 0.229783E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
3SED= 0.158641E+01 Area = 0.120401E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
4SED= 0.125731E+01 Area = 0.253623E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
5SED= 0.174745E+01 Area = 0.232987E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
6SED= 0.171150E+01 Area = 0.200962E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
7SED= 0.126023E+01 Area = 0.253154E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
8SED= 0.126401E+01 Area = 0.251216E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
9SED= 0.178617E+01 Area = 0.269591E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
10SED= 0.166056E+01 Area = 0.163283E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
11SED= 0.126244E+01 Area = 0.248240E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
12SED= 0.126183E+01 Area = 0.247641E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
13SED= 0.178479E+01 Area = 0.268204E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
14SED= 0.159552E+01 Area = 0.125550E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
15SED= 0.126263E+01 Area = 0.249410E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
16SED= 0.125806E+01 Area = 0.250174E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
17SED= 0.174629E+01 Area = 0.231954E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
18SED= 0.166023E+01 Area = 0.163087E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
19SED= 0.125602E+01 Area = 0.251812E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
20SED= 0.171075E+01 Area = 0.200384E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
21SED= 0.158436E+01 Area = 0.119358E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13
22SED= 0.174452E+01 Area = 0.228787E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13

23SED= 0.175955E+01 Area = 0.243884E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155075E+01 N= 13

1SED= 0.172386E+01 Area = 0.247786E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
2SED= 0.171200E+01 Area = 0.232060E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
3SED= 0.158584E+01 Area = 0.120437E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
4SED= 0.131680E+01 Area = 0.247867E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
5SED= 0.171333E+01 Area = 0.235320E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
6SED= 0.168515E+01 Area = 0.202553E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
7SED= 0.131965E+01 Area = 0.247466E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
8SED= 0.132328E+01 Area = 0.245646E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
9SED= 0.174377E+01 Area = 0.272888E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
10SED= 0.164482E+01 Area = 0.164079E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
11SED= 0.132195E+01 Area = 0.242705E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
12SED= 0.132142E+01 Area = 0.242108E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
13SED= 0.174270E+01 Area = 0.271463E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
14SED= 0.159269E+01 Area = 0.125659E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
15SED= 0.132211E+01 Area = 0.243853E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
16SED= 0.131784E+01 Area = 0.244511E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
17SED= 0.171241E+01 Area = 0.234261E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
18SED= 0.164455E+01 Area = 0.163879E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
19SED= 0.131572E+01 Area = 0.246072E+00 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
20SED= 0.168456E+01 Area = 0.201961E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
21SED= 0.158419E+01 Area = 0.119378E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
22SED= 0.171110E+01 Area = 0.231039E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14
23SED= 0.172279E+01 Area = 0.246496E+01 exlk= 0.100055E+01 SSP= 0.100000E+00 SEDavg= 0.155489E+01 N= 14

1SED= 0.169158E+01 Area = 0.250033E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
2SED= 0.168234E+01 Area = 0.234002E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
3SED= 0.158239E+01 Area = 0.120519E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
4SED= 0.136380E+01 Area = 0.243468E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
5SED= 0.168333E+01 Area = 0.237309E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
6SED= 0.166121E+01 Area = 0.203926E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
7SED= 0.136651E+01 Area = 0.243126E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
8SED= 0.136989E+01 Area = 0.241404E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
9SED= 0.170733E+01 Area = 0.275679E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
10SED= 0.162926E+01 Area = 0.164792E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
11SED= 0.136880E+01 Area = 0.238490E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
12SED= 0.136834E+01 Area = 0.237894E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
13SED= 0.170649E+01 Area = 0.274222E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
14SED= 0.158748E+01 Area = 0.125798E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
15SED= 0.136893E+01 Area = 0.239621E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
16SED= 0.136502E+01 Area = 0.240190E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
17SED= 0.168260E+01 Area = 0.236228E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
18SED= 0.162904E+01 Area = 0.164588E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
19SED= 0.136291E+01 Area = 0.241684E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
20SED= 0.166074E+01 Area = 0.203323E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
21SED= 0.158107E+01 Area = 0.119447E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
22SED= 0.168164E+01 Area = 0.232961E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15
23SED= 0.169074E+01 Area = 0.248716E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155615E+01 N= 15

1SED= 0.166550E+01 Area = 0.251918E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
2SED= 0.165828E+01 Area = 0.235637E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
3SED= 0.157896E+01 Area = 0.120620E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
4SED= 0.140201E+01 Area = 0.240076E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
5SED= 0.165901E+01 Area = 0.238980E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
6SED= 0.164160E+01 Area = 0.205091E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
7SED= 0.140453E+01 Area = 0.239786E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
8SED= 0.140761E+01 Area = 0.238147E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
9SED= 0.167801E+01 Area = 0.278014E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
10SED= 0.161623E+01 Area = 0.165412E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
11SED= 0.140671E+01 Area = 0.235254E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
12SED= 0.140632E+01 Area = 0.234657E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
13SED= 0.167734E+01 Area = 0.276532E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
14SED= 0.158273E+01 Area = 0.125944E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
15SED= 0.140682E+01 Area = 0.236371E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
16SED= 0.140331E+01 Area = 0.236865E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
17SED= 0.165844E+01 Area = 0.237882E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
18SED= 0.161605E+01 Area = 0.165205E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
19SED= 0.140128E+01 Area = 0.238302E+00 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
20SED= 0.164122E+01 Area = 0.204479E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
21SED= 0.157790E+01 Area = 0.119537E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
22SED= 0.165772E+01 Area = 0.234578E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16
23SED= 0.166484E+01 Area = 0.250578E+01 exlk= 0.100015E+01 SSP= 0.100000E+00 SEDavg= 0.155706E+01 N= 16

1SED= 0.164488E+01 Area = 0.253483E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
2SED= 0.163922E+01 Area = 0.236998E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
3SED= 0.157617E+01 Area = 0.120723E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
4SED= 0.143329E+01 Area = 0.237442E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
5SED= 0.163976E+01 Area = 0.240371E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
6SED= 0.162601E+01 Area = 0.206067E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
7SED= 0.143559E+01 Area = 0.237199E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
8SED= 0.143835E+01 Area = 0.235628E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
9SED= 0.165484E+01 Area = 0.279951E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
10SED= 0.160582E+01 Area = 0.165940E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
11SED= 0.143761E+01 Area = 0.232751E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
12SED= 0.143727E+01 Area = 0.232155E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
13SED= 0.165431E+01 Area = 0.278447E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
14SED= 0.157892E+01 Area = 0.126082E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
15SED= 0.143770E+01 Area = 0.233859E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
16SED= 0.143459E+01 Area = 0.234288E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
17SED= 0.163931E+01 Area = 0.239258E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
18SED= 0.160568E+01 Area = 0.165731E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
19SED= 0.143270E+01 Area = 0.235676E+00 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17

20SED= 0.162571E+01 Area = 0.205447E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
21SED= 0.157532E+01 Area = 0.119631E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
22SED= 0.163877E+01 Area = 0.235925E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17
23SED= 0.164435E+01 Area = 0.252124E+01 exlk= 0.100027E+01 SSP= 0.100000E+00 SEDavg= 0.155809E+01 N= 17

1SED= 0.162921E+01 Area = 0.254772E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
2SED= 0.162476E+01 Area = 0.238121E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
3SED= 0.157456E+01 Area = 0.120821E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
4SED= 0.145939E+01 Area = 0.235386E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
5SED= 0.162517E+01 Area = 0.241518E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
6SED= 0.161429E+01 Area = 0.206876E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
7SED= 0.146145E+01 Area = 0.235183E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
8SED= 0.146389E+01 Area = 0.233670E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
9SED= 0.163718E+01 Area = 0.281544E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
10SED= 0.159819E+01 Area = 0.166384E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
11SED= 0.146328E+01 Area = 0.230805E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
12SED= 0.146300E+01 Area = 0.230208E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
13SED= 0.163676E+01 Area = 0.280023E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
14SED= 0.157656E+01 Area = 0.126206E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
15SED= 0.146336E+01 Area = 0.231905E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
16SED= 0.146064E+01 Area = 0.232281E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
17SED= 0.162481E+01 Area = 0.240393E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
18SED= 0.159807E+01 Area = 0.166172E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
19SED= 0.145891E+01 Area = 0.233625E+00 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
20SED= 0.161405E+01 Area = 0.206250E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
21SED= 0.157388E+01 Area = 0.119721E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
22SED= 0.162441E+01 Area = 0.237037E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18
23SED= 0.162879E+01 Area = 0.253398E+01 exlk= 0.100063E+01 SSP= 0.100000E+00 SEDavg= 0.155976E+01 N= 18

1SED= 0.161363E+01 Area = 0.255827E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
2SED= 0.161013E+01 Area = 0.239042E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
3SED= 0.157019E+01 Area = 0.120908E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
4SED= 0.147766E+01 Area = 0.233774E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
5SED= 0.161043E+01 Area = 0.242458E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
6SED= 0.160182E+01 Area = 0.207542E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
7SED= 0.147950E+01 Area = 0.233605E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
8SED= 0.148163E+01 Area = 0.232141E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
9SED= 0.161999E+01 Area = 0.282848E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
10SED= 0.158899E+01 Area = 0.166752E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
11SED= 0.148115E+01 Area = 0.229285E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
12SED= 0.148091E+01 Area = 0.228688E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
13SED= 0.161965E+01 Area = 0.281313E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
14SED= 0.157163E+01 Area = 0.126313E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
15SED= 0.148120E+01 Area = 0.230379E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
16SED= 0.147885E+01 Area = 0.230709E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
17SED= 0.161014E+01 Area = 0.241324E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
18SED= 0.158890E+01 Area = 0.166539E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
19SED= 0.147727E+01 Area = 0.232017E+00 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
20SED= 0.160163E+01 Area = 0.206911E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
21SED= 0.156965E+01 Area = 0.119802E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
22SED= 0.160984E+01 Area = 0.237948E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19
23SED= 0.161329E+01 Area = 0.254441E+01 exlk= 0.100004E+01 SSP= 0.100000E+00 SEDavg= 0.155818E+01 N= 19

1SED= 0.160415E+01 Area = 0.256686E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
2SED= 0.160139E+01 Area = 0.239793E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
3SED= 0.156953E+01 Area = 0.120983E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
4SED= 0.149516E+01 Area = 0.232504E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
5SED= 0.160162E+01 Area = 0.243225E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
6SED= 0.159477E+01 Area = 0.208086E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
7SED= 0.149676E+01 Area = 0.232364E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
8SED= 0.149861E+01 Area = 0.230942E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
9SED= 0.160925E+01 Area = 0.283910E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
10SED= 0.158452E+01 Area = 0.167055E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
11SED= 0.149821E+01 Area = 0.228093E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
12SED= 0.149801E+01 Area = 0.227495E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
13SED= 0.160899E+01 Area = 0.282363E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
14SED= 0.157054E+01 Area = 0.126403E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
15SED= 0.149826E+01 Area = 0.229182E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
16SED= 0.149624E+01 Area = 0.229474E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
17SED= 0.160138E+01 Area = 0.242082E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
18SED= 0.158444E+01 Area = 0.166841E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
19SED= 0.149484E+01 Area = 0.230750E+00 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
20SED= 0.159462E+01 Area = 0.207451E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
21SED= 0.156909E+01 Area = 0.119873E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
22SED= 0.160116E+01 Area = 0.238692E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20
23SED= 0.160388E+01 Area = 0.255291E+01 exlk= 0.100048E+01 SSP= 0.100000E+00 SEDavg= 0.155980E+01 N= 20

1SED= 0.159399E+01 Area = 0.257384E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
2SED= 0.159181E+01 Area = 0.240403E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
3SED= 0.156642E+01 Area = 0.121047E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
4SED= 0.150678E+01 Area = 0.231500E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
5SED= 0.159198E+01 Area = 0.243846E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
6SED= 0.158655E+01 Area = 0.208529E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
7SED= 0.150819E+01 Area = 0.231386E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
8SED= 0.150978E+01 Area = 0.229998E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
9SED= 0.159808E+01 Area = 0.284772E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
10SED= 0.157836E+01 Area = 0.167302E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
11SED= 0.150947E+01 Area = 0.227155E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
12SED= 0.150930E+01 Area = 0.226557E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
13SED= 0.159787E+01 Area = 0.283215E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
14SED= 0.156713E+01 Area = 0.126479E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
15SED= 0.150949E+01 Area = 0.228240E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
16SED= 0.150776E+01 Area = 0.228500E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21

```
17SED= 0.159179E+01 Area = 0.242697E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
18SED= 0.157830E+01 Area = 0.167087E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
19SED= 0.150652E+01 Area = 0.229750E+00 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
20SED= 0.158643E+01 Area = 0.207890E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
21SED= 0.156607E+01 Area = 0.119933E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
22SED= 0.159163E+01 Area = 0.239295E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
23SED= 0.159378E+01 Area = 0.255980E+01 exlk= 0.100006E+01 SSP= 0.100000E+00 SEDavg= 0.155859E+01 N= 21
```

fea3d2_out.txt

```
 istep   exlk      appliedforce     a(28)     a(38)     it  Weight    Area1     Area2     Area3     Area4    xlie 10    xlie 15     fin1
*   1  0.100000E+00   -0.60E+02  0.1490-313  0.1534-313   3 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312449E+02 0.287421E+02 -.36E+03
*   2  0.163377E+00   -0.98E+02   -.3916E-01   -.4494E-01   3 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312445E+02 0.287418E+02 -.59E+03

*  85  0.287754E+00   -0.17E+03   -.1057E+00   -.1212E+00   2 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312439E+02 0.287412E+02 -.11E+04

* 202  0.375905E+00   -0.23E+03   -.1385E+00   -.1589E+00   2 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312434E+02 0.287407E+02 -.14E+04

* 396  0.514589E+00   -0.31E+03   -.1905E+00   -.2186E+00   2 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312426E+02 0.287400E+02 -.19E+04

* 579  0.639982E+00   -0.38E+03   -.2379E+00   -.2729E+00   2 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312420E+02 0.287394E+02 -.24E+04

* 775  0.770842E+00   -0.46E+03   -.2877E+00   -.3300E+00   2 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312412E+02 0.287387E+02 -.28E+04

* 968  0.897685E+00   -0.54E+03   -.3364E+00   -.3859E+00   2 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312405E+02 0.287381E+02 -.33E+04

*1125  0.100006E+01   -0.60E+03   -.3760E+00   -.4312E+00   1 0.119850E+03 0.257384E+01 0.240403E+01 0.121047E+01 0.231500E+00 0.312400E+02 0.287375E+02 -.37E+04
```

APPENDIX C-FORTRAN Source Code

The source code for testing the arc length method is given in C.1. This code was used to run the function tests given in section 2.2 and the 3D truss examples in chapter IV. The source listing in C.2 is for the proposed Hrinda optimization that was used for the test problems in chapter VIII.

C.1 Hrinda Rikarc

```
c**********************************************************************
c    Hrinda FEM using arc length
c**********************************************************************
     implicit real*8(a-h,o-z)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
c
     dimension nel(6),ndofpe(6),iflag(100)  ! assuming 6 finite element types
c                    dimension esin(6,6),elk(6,6)
c                    dimension ux(2),uy(2)

     common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
     common/disp/ux(2),uy(2),uz(2),P(6),B1(6)

                         common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
    $    dqk2(5000),fintk(5000),
    $    resk(5000),scali(5000),work1(5000),indx(5000),neq

c    common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)
c
     integer, allocatable:: ie(:),je(:),iboundc(:),iet(:),jet(:)
    $,ip(:),iu(:),iut(:),iup(:),isupn(:),lm(:),iperm(:),ia(:),ja(:)
c    $,ip(:),iu(:),iut(:),iup(:),isupn(:),lm(:),iperm(:),ia(:)
c    $,ip(:),iu(:),iut(:),iup(:),isupn(:),lm(:),iperm(:),ia(:)
c    $,ju(:),jut(:),itempo1(:),ia1(:),ia2(:),exfor(:)
    $,ju(:),jut(:),itempo1(:)

     double precision, allocatable:: b(:),be(:),ae(:),ad(:)
c    $,elk(:,:),x(:),y(:),z(:),propmat(:),propsect(:)
    $,x(:),y(:),z(:),propmat(:),propsect(:)
c    $,x(:),y(:),z(:),propmat(:),propsect(:)
    $,tempo1(:),tempo2(:)
    $,di(:),an(:),un(:),evalues(:),evectors(:),dm(:),am(:)
    $,elm(:,:)
    $,disp(:),qk(:),exfor(:)

c    $,ux(:),uy(:),finx(:),finy(:)
     double precision, allocatable:: tempo3(:)
c

c                    open(unit=5,file='input3.dat',status='old')
c                    open(unit=6,file='output3.dat',status='old')

cbar                 open(unit=5,file='bar2.dat',status='old')
carch3d              open(unit=5,file='arch3din.dat',status='old')
carch2               open(unit=5,file='arch2in.dat',status='old')
carch3d              open(unit=5,file='arch3din.dat',status='old')
cKhot                open(unit=5,file='Khot_shallow_truss_in.dat',status='old')

cframe               open(unit=5,file='frame_in.dat',status='old')
cframe2              open(unit=5,file='frame2_in.dat',status='old')

ct3a                 open(unit=5,file='t3a_in.dat',status='old')!3mem truss triangle

cpod                 open(unit=5,file='tpod_in.dat',status='old')
c                    open(unit=5,file='tpod2_in.dat',status='old')
ctwo                 open(unit=5,file='twobar_in.dat',status='old')
cdome                open(unit=5,file='dome_in.dat',status='old')
csnap                open(unit=5,file='snap_in.dat',status='old')

csnap4               open(unit=5,file='snap4_in.dat',status='old')

c                    open(unit=5,file='snap5_in.dat',status='old')

cdome7               open(unit=5,file='dome7_in.dat',status='old')
ctruss               open(unit=5,file='truss7_3d_in.dat',status='old')

c                    open(unit=5,file='bar2a.dat',status='old')
c                    open(unit=5,file='bar2b.dat',status='old')
cbar                 open(unit=6,file='bar2out.dat',status='old')
carch                open(unit=6,file='arch3dot.dat',status='old')
cframe               open(unit=6,file='frame_out.dat',status='old')
ctwo                 open(unit=6,file='twobar_out.dat',status='old')
cdome                open(unit=6,file='dome_out.dat',status='old')
csnap                open(unit=6,file='snap_out.dat',status='old')
ct3a                 open(unit=6,file='t3a_ot.dat',status='old')
cpod                 open(unit=6,file='tpod_ot.dat',status='old')
ctruss               open(unit=6,file='truss7_3d_ot.dat',status='old')
c                    open(unit=7,file='data_out.dat',status='old')

cfunc                open(unit=5,file='func_in.dat',status='old')
cOK                  open(unit=5,file='func_in_Aora_2.dat',status='old')
                     open(unit=5,file='func_in_Aora_0.dat',status='old')
cOK                  open(unit=5,file='func_in_Hrinda_1.dat',status='old')
c                    open(unit=5,file='func_in_Hrinda_2.dat',status='old')
cOK                  open(unit=5,file='func_in_Hrinda_3.dat',status='old')
c                    open(unit=5,file='func_in_JDennis_1.dat',status='old')

c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c
     write(6,*) 'Version Date = 03-01-2004'
c.....input general information
                     iex=10 !Function Testing
c                    icxample=2
     neltype=6             ! assuming 6 finite element types
                     ityp=12 ! type elc is 3d from GAH

c1000  call funct(iexample,u,f,fd,neq,iforce,L1,L1i)

     call generalip(nel,ndofpe,neltype,maxdofpe,nboundc
    $,          ndofpn,numnodes,loadof,nummat,nlayers,nsect
    $,          lumpmass,ndof,ieall,jeall,ianal,neig,iarc
    $,          itype)
c                    print *,'ndof=',ndof
c                    pause
c                    print *,'ianal=',ianal
c                    print *,'lumpmass=',lumpmass
```

```
c                     print *,'nel=',nel
c                     stop
c                      print *,'maxdofpe=',maxdofpe
c                     print *,'neltype=',neltype
c                     print *,'itype=',ityype
c                     print *,'iarc=',iarc
c                     pause
c                     print *,'itruss=',itruss
c                     stop
c.....dynamic storage memory allocation
      allocate ( ie(ieall+1),je(jeall),iboundc(ndof),iet(ndof+1),
     $         jet(jeall),ip(ndof),iu(ndof+1),iut(ndof+1),iup(ndof),
     $         isupn(ndof),lm(maxdofpe),iperm(ndof),ia(ndof+1))
c     $          ia1(ndof+1),ia2(ndof+1) )
cc                    allocate (ja(50))
cc                    allocate (an(100))
c
      allocate ( b(ndof),be(maxdofpe),ae(maxdofpe**2),ad(ndof),
c     $       an(ndof),
     $      an(10000),
c     $       elk(maxdofpe,maxdofpe)
     $      x(numnodes),y(numnodes),z(numnodes),
     $      propmat(nummat*11),propsect(nsect*12),tempo1(ndof),
     $      tempo2(ndof),di(ndof),exfor(ndof))

      allocate ( disp(ndof),qk(ndof) )

c.....input material properties
c                     print *,'Check before materprop'
      call materprop(nummat,propmat,youngm)
c                     print *,'Check after materprop'
c.....input cross-sectional properties
      call sectprop(nsect,propsect,area)
c.....input joint coordinates

c                     print *,'youngm=',youngm
c                     print *,'area=',area
c                     stop

      call nodecoor(numnodes,x,y,z)

c                     print *,'x & y & z =',x,' ',y,' ',z
c                     stop

c     print *,'neltype = ',neltype
c                     print *,'nel = ',nel
c                     print *,'ndofpe = ',ndofpe
c                     print *,'ndofpn = ',ndofpn
c                     print *,'before elconect'
c                     pause
c.....input connectivity information
      call elconect(ie,je,lm,neltype,nel,ndofpe,ndofpn)

c                     print *,'ie after elconect =',(ie(i),i=1,ieall+1)
c                     print *,'je after elconect =',(je(i),i=1,ieall+1)
c                     pause
c                     stop

c.....input applied loads at the joints
      call loads(ndof,b,loadof,exfor)
c                     print *,'exfor =',(exfor(i),i=1,ndof)
c                     pause
c                     print *,'b after loads=',(b(i),i=1,ndof)
c                     pause
c                     stop
c.....input boundary dof
c                     print *,'chk before supportdof'
c                     stop
      call supportdof(iboundc,nboundc,b,ndof,ia)
c     call supportdof(iboundc,nboundc,b,ndof,ia1)
c                     print *,'iboundc=',(iboundc(i),i=1,ndof)
c                     print *,'chk before call rikarc'
c                     print *,'iarc before rikarc=',iarc
c                     stop
c                     print *,'ia after supportdof =',(ia(i),i=1,ndof+1)
c                     print *,'ia2 before rikarc=',(ia2(i),i=1,ndof+1)
c                     print *,'b before rikarc=',(b(i),i=1,ndof)
c                     pause
c                     stop

c                     Do 10 i=1,ndof+1
c10                   ia2(i)=ia1(i)

c                     print *,'ia2(_)=',(ia2(i),i=1,ndof+1)
c                     stop
c**Arc Length**
                      if(iarc .eq. 1)then
c                     print *,'<<<<<<<<<<call to rikarc>>>>>>>>>>'
c                     pause
c                                   print *,'x & y & z =',x,' ',y,' ',z
c                     stop
                      call rikarc(ie,je,lm,neltype,nel,ndofpe,ndofpn,
     $  numnodes,x,y,z,
     $  ndof,b,loadof,
     $  iboundc,nboundc,ia,
c     $  iboundc,nboundc,ia1,ia2,
     $  ieall,ji,iet,jet,
     $  itempo1,exfor,
c     $  ja,
     $  ip,ad,an,propmat,propsect,be, !call assembly
c     $  ip,ad,an,propmat,propsect,elk,be, !call assembly
     $  ae,maxdofpe,
     $  ianal,lumpmass,dm,am,elm,tempo1,
     $  iperm,  !call minfills
     $  m,ncoef2,ju,  !call symfactd
     $  iu,iut,  !call transad
     $  isupn, !call supernode
     $  di,un,iup,iopf,  !call numfa1d
     $  iopb,  !call fbed
     $  disp,iarc,jeall,ux,uy,
     $  qk,zee,dLength,iex)
c     $  ja,qk)
c     $  ja,csm,elk,qk)

                      endif
c                     print *,'STOP after RIK'
c                     stop
c**

c.....
c                     print *,'ie before transa2d =',(ie(i),i=1,ieall+1)
c                     print *,'je before transa2d =',(je(i),i=1,jeall)
c                     print *,'iet after transa2d =',(jet(i),i=1,ieall+1)
c                     print *,'jet after transa2d =',(jet(i),i=1,jeall)
c                     print *,'chk before transa2d'
c                     pause
c                     stop
      call transa2d(ieall,ndof,ie,je,iet,jet)
      write(6,*) '*** passed transa2 ***'
c                     print *,'*** passed transa2 ***'
c                     print *,'ieall = ',ieall
c                     print *,'
c     print *,'ie = ',ie
c                     print *,'je = ',je
c                     print *,'iet = ',iet
c                     print *,'jet = ',jet
c                     stop
c.....sparse "symbolic" assembly
      ncoef1_guess=20000000
      allocate ( itempo1(ncoef1_guess) )
c
c                     print *,'ia1 before sysbass= ',(ia1(i),i=1,ndof+1)
c                     print *,'ia2 before sysbass= ',(ia2(i),i=1,ndof+1)
c                     stop
      call symbass(ie,je,iet,jet,ndof,ia,
     $            itempo1,ncoef1)
c     call symbass(ie,je,iet,jet,ndof,ia1,ia2,
c     $            itempo1,ncoef1)
c                     print *,'*** After symbass ***'
c                     print *,'ieall = ',ieall
c     print *,'ic = ',ie
c                     print *,'je = ',je
c                     print *,'iet = ',iet
c                     print *,'jet = ',jet
c                     print *,'ia after symbass= ',(ia(i),i=1,ndof+1)
c                     print *,'ia2 after sysbass= ',(ia2(i),i=1,ndof+1)
c                     pause
cc                    do 111 i=1,19
cc111                 print *,'itempo1 = ',itempo1(i)
cc                    stop
c                     print *,'ncoef1 = ',ncoef1(i)
c     stop
c
                      print *,'chk after call copy_int'
      allocate ( ja(ncoef1) )
                      print *,'chk after allocate ja'
c      allocate ( an(ncoef1) )
                      print *,'Chk before copy_int'
      call copy_int(ncoef1,itempo1,ja)
      deallocate (itempo1)
c                     print *,'chk after call copy_int'
c                     stop
c
      write(6,*) '*** passed symbass ***'
c.....sparse "numerical" assembly
      if (ianal .eq. 1) then
       allocate ( evalues(neig) )
       allocate ( evectors(ndof*neig) )
       allocate ( dm(ndof) )
       allocate ( elm(maxdofpe,maxdofpe) )
       if (lumpmass .eq. 0) then
       allocate ( am(ncoef1) )
       endif
      endif
c
      write(6,*) 'check point #01'
c                     print *,'chk before assembly'
c                     stop
c                     call assembly(ip,ad,an,lm,ie,je,propmat
c     $,          propsect,elk,be,b,iboundc,ux,uy
c     $,          ae,ia2,ja,x,y,ncoef1,maxdofpe,
c     $ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,tempo1,disp,
c     $iarc,ityp,getforce,it)

      call assembly(ip,ad,an,lm,ie,je,propmat,
     $          propsect,be,b,iboundc,
c     $          propsect,elk,be,b,iboundc,
     $          ae,ia,ja,x,y,z,ncoef1,maxdofpe,
c     $          ae,ia2,ja,x,y,z,ncoef1,maxdofpe,
     $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c     $ tempo1,disp,iarc,it,area,youngm,esm,qk,itnrmlon)
     $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon,istep,icheck,
     $ zee,dLength,iex)

c                      print *,'chk after assembly'
c                     print *,'ae = ',(ae(i),i=1,maxdofpe**2)
                      write(6,*)'ae=',(ae(i),i=1,maxdofpe*2)
                      write(6,*)'disp=',(disp(i),i=1,ndof)
c                     write(6,*)'ae=',(ae(i),i=1,8)
c                     write(6,*)'disp=',(disp(i),i=1,10)
c                      print *,'ac = ',ae
c                     stop

      write(6,*) 'check point #02'
      if (ianal .eq. 1) then
      write(6,*) 'diag-mass = dm(-) = ',(dm(i),i=1,ndof)
      endif
      if (ianal .eq. 1  .and.  lumpmass .eq. 0) then
      write(6,*) 'offdiag-mass = am(-) = ',(am(i),i=1,ncoef1)
      endif
c
c.....assuming to skip the "REORDERING" process (minimize "fills-in" terms)
```

```
c
      call minfills(ndof,iperm)
c......eigen-solution
      if (ianal .eq. 1) then
      lump=lumpmass
      ishift=0
      iprint=1
      mtot=20000000
      allocate( tempo3(mtot) )
c
c     r(mtot)=working array (in real*8) = tempo3(-)
c     iflag(1,2,3,4,5,6,7,8,9,10)=nreord,loop_unroll,??,??,neig
c                       =iprint,lump,ishift,ncoef2,??
c     nreord=0,1,2,3=no reorder,metis,nd,mmd
c
      iflag(1)=0        ! 1,2,3 = metis,nd,mmd
      iflag(2)=1        ! level of unroll
      iflag(5)=neig
      iflag(6)=iprint
      iflag(7)=lump
      iflag(8)=ishift
c     iflag(9)=ncoef2     ! as an output
c
      write(6,*) 'inside *cmei*.f: ndof,ncoef1,neig,lump = '
      write(6,*)              ndof,ncoef1,neig,lump
      write(6,*) 'inside *cmei*.f: ishift,mtot = '
      write(6,*)              ishift,mtot
      write(6,*) 'inside *cmei*.f: iflag(10) = ',(iflag(i),i=1,10)
c      write(6,*) 'inside *cmei*.f: ia(-) = ',(ia(i),i=1,ndof+1)
      write(6,*) 'inside *cmei*.f: ja(-) = ',(ja(i),i=1,ncoef1)
      write(6,*) 'inside *cmei*.f: ad(-) = ',(ad(i),i=1,ndof)
      write(6,*) 'inside *cmei*.f: an(-) = ',(an(i),i=1,ncoef1)
      write(6,*) 'inside *cmei*.f: dm(-) = ',(dm(i),i=1,ndof)
c
ccc      if (lump .eq. 1) then
ccc      call eigsolver011(ndof,ncoef1,neig,lump,ishift,
ccc     $mtot,ia,ja,ad,an,dm,   evalues,evectors,tempo3,iflag,iperm)
ccc      elseif (lump .eq. 0) then
ccc      call eigsolver022(ndof,ncoef1,neig,lump,ishift,
ccc     $mtot,ia,ja,ad,an,dm,am,evalues,evectors,tempo3,iflag,iperm)
ccc      endif
      deallocate( tempo3 )
      STOP
      endif
c......sparse "symbolic" factorization
      ncoef2_guess=40000000
      allocate ( itempo1(ncoef2_guess) )

      call symfactd(ndof,ia,ja,iu,itempo1,ip,ncoef2)
c      call symfactd(ndof,ia2,ja,iu,itempo1,ip,ncoef2)

      write(6,*) 'check point #03'
                   write(6,*)'iu after symfactd=',(iu(i),i=1,ndof+1)
                   write(6,*)'itempo1 after symfactd=',(itempo1(i),i=1,25)
                   write(6,*)'ip after symfactd=',(ip(i),i=1,ndof)
c                  stop


c                  print *,'ncoef2=',ncoef2
c                  stop
c
      allocate ( ju(ncoef2) )
      allocate ( jut(ncoef2) )
      allocate ( un(ncoef2) )
       call copy_int(ncoef2,itempo1,ju)
       deallocate (itempo1)
c
c......transpose twice to put column numbers in order
      call transad(ndof,ndof,iu,ju,iut,jut)
c                  print *,'*** passed transa ***'
      write(6,*) '*** passed transa ***'
c
c......assuming to skip the "SUPER NODE" process (for unrolling purpose)
c
      call supernode(ndof,isupn)
c
c......sparse "numerical" factorization
      call numfa1d(ndof,ia,ja,ad,an,iu,
     $          ju,di,un,ip,iup,
     $          isupn,iopf)
c routine numfa1d(n,ia,ja,ad,an,iu,ju,di,un,ip,iup,isupd,iopf)
c      call numfa1d(ndof,ia2,ja,ad,an,iu,
c    $          ju,di,un,ip,iup,
c    $          isupn,iopf)
      write(6,*) '*** passed numfa1 ***'

c                   print *,'b after numfa1d=',(b(i),i=1,ndof)

c                   write(6,*)'n before fbed=',n
                    write(6,*)'b before fbed=',(b(i),i=1,ndof)
                    write(6,*)'di before fbed=',(di(i),i=1,ndof)
                    write(6,*)'iu before fbed=',(iu(i),i=1,ndof+1)
                    write(6,*)'ju before fbed=',(ju(i),i=1,ncoef2)
                    write(6,*)'un before fbed=',(un(i),i=1,ncoef2)

c                   print *,'di before fbed=',(di(i),i=1,ndof)
c                   print *,'iu before fbed=',(iu(i),i=1,ndof+1)
c                   print *,'ju before fbed=',(ju(i),i=1,ncoef2)
c                   print *,'ncoef2=',ncoef2
c                   pause
c                  stop

c......sparse forward/backward solution phase
      call fbed(ndof,iu,ju,di,un,b,
     $        tempo1,iopb,isupn,getsolve)

c     print *, '*** passed fbe ***'
c                   print *, 'tempo1(i)=',(tempo1(i),i=1,ndof)
                    do 50 i=1,ndof
                   print *,'disp = ',i,' - ',tempo1(i)
50      disp(i)=tempo1(i)
c                    write(6,*)'disp(i)=',(disp(i),i=1,ndof)

c                    print *,'tempo1, disp = ',tempo1
c                   print *,'b = ',b
                   stop

      write(6,*) '*** passed fbe ***'
c......error norm check
      call mvspar(ad,ja,an,tempo1,ia,
     $          ndof,ncoef1,tempo2,ioper,isupn)
      write(6,*) '*** passed mvspar ***'
      call errnormd(tempo2,b,tempo1,disamax,locate,
     $          disasum,ndof,relerr)
      write(6,*) '*** passed ernorm ***'
c
c......assuming to skip the "TRANSFORMATION BACK" to original solution
c......due to REORDERING
c
c     return
      stop
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine elmass(rho,area,xl,lump,elm,ndofpe,maxdofpe,
     $cx,cy)
      implicit real*8(a-h,o-z)
c......get the 2-d truss element mass matrices
      dimension elm(maxdofpe,1)
      const=rho*area*xl
      do 1 i=1,ndofpe
       do 2 j=1,ndofpe
       elm(i,j)=0.
 2      continue
 1     continue
c
      if (lump .eq. 1) then    ! lumped mass is used
      do 3 i=1,ndofpe
      elm(i,i)=const/2.0
 3     continue
      elseif (lump .eq. 0) then    ! consistent mass is used
c
c     [m] = [lamdat] * [m_local] * [lamda]  -->> done by MATLAB
c     where:
c     [lamda] = [ cx  cy   0   0       ----->> Gere & Weaver's book
c              -cy  cx   0   0              page 255
c                0   0  cx  cy
c                0   0 -cy  cx ]
c     [m_local] = (rho*area*xl/6) * [ 2   0   1   0    -->> J.N. Reddy's book
c                                0   0   0   0        page 233
c                                1   0   2   0
c                                0   0   0   0 ]
      const=const/6.0
      cxcx=cx*cx
      cycy=cy*cy
      cxcy=cx*cy
c
      elm(1,1)=const*cxcx*2.0
      elm(1,2)=const*cxcy*2.0
      elm(1,3)= elm(1,1)/2.0
      elm(1,4)= elm(1,2)/2.0
c
      elm(2,2)=const*cycy*2.0
      elm(2,3)=elm(1,4)
      elm(2,4)= elm(2,2)/2.0
c
      elm(3,3)=elm(1,1)
      elm(3,4)=elm(1,2)
c
      elm(4,4)=elm(2,2)
c
      do 12 i=1,ndofpe
       do 13 j=i,ndofpe
       elm(j,i)=elm(i,j)
 13     continue
 12    continue
c
      endif
       write(6,*) 'inside elmass: elm(i,i)= ',(elm(i,i),i=1,ndofpe)
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine elstif(ndofpe,esm,be,
     $          area,youngm,xlength,cx,cy,cz,maxdofpe,
     $          xl,xli,deltaxi,deltayi,deltazi,x,y,z,
     $       iarc,xi,xj,yi,yj,zi,zj,ityp,lm,istep,iel,icheck,iex)

      implicit real*8(a-h,o-z)

c      common/elmatx/esm(6,6),ef(6),x(6),y(6),z(6),kl,ndofpe
      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/disp/ux(2),uy(2),uz(2),P(6),B1(6)
c      common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)

c......get the 2-d truss element stiffness, and element load vector !
      dimension be(*),esm(6,6),lm(1)
c      dimension be(*),esm(maxdofpe,*),lm(1)

                  dimension C(6)
c                 dimension P(6), B1(6)

c                 dimension x(3),y(3),z(3)
                  dimension x(*),y(*),z(*)

c                 print *,'iel at start elstif=',iel !this is the element number
c                 print *,'xli at start elstif=',xli !total length of element updated
c                 print *,'lm1 at start elstif=',(lm(ic),ic=1,6)
c                 pause

                  icheck=0 !gah use to chk big stiff matrix when iel=4 with 2" axial disp.

                  If(iex .eq. 10)then
c**Function Test**
C***Crisfield sdof
```

```
c      AE=5.e7
c      xl=2500.125
c      stifi(1,1) = (AE/xl**3.)*(625.d0+75.d0*u1+
c     $            1.5d0*(u1**2.))

C***Arora's nonlinear 1
c      stifi(1,1) = 3.d0*u1+10.d0
c      stifi(1,2) = 12.d0*u2
c      stifi(2,1) = 24.d0*u2
c      stifi(2,2) = 4.d0

C***Arora's nonlinear 2  soln=(-.33, -.5)   **OK**
c      esm(1,1) = 24.d0
c      esm(1,2) = -12.d0
c      esm(2,1) = -12.d0
c      esm(2,2) = 8.d0

C***J. Dennis's opt. book page 172
c      stifi(1,1) = 1.d0
c      stifi(1,2) = 1.d0
c      stifi(2,1) = u1
c      stifi(2,2) = u2

C***Auroa's soln=(0,0)     **OK**
       esm(1,1) = 10.d0
       esm(1,2) = 2.d0
       esm(2,1) = 2.d0
       esm(2,2) = 2.d0

C***J. Dennis opt soln=(0.000033233553246,0.000033233553246)  **OK
c      esm(1,1) = 4.d0*ux(1)*ux(1)+1.d-10
c      esm(1,2) = 4.d0*ux(1)*uy(1)+2.d-10
c      esm(2,1) = 4.d0*ux(1)*uy(1)+3.d-10
c      esm(2,2) = 4.d0*uy(1)*uy(1)+.d-10

c      esm(1,1) = 4.d0*ux(1)*ux(1) + 2.d-6
c      esm(1,2) = 4.d0*ux(1)*uy(1) + 1.d-6
c      esm(2,1) = 4.d0*ux(1)*uy(1) + 1.d-6
c      esm(2,2) = 4.d0*uy(1)*uy(1) + 2.d-6

c***Hrinda 1 (0.,20.)  OK  **OK**(-.7633, 1.165)
cc      esm(1,1) = 24.d0*ux(1) + 2.d-6
cc      esm(1,2) = -12.d0 + 1.d-6
cc      esm(2,1) = -12.d0 + 1.d-6
cc      esm(2,2) = 8.d0*uy(1) + 2.d-6

c***Hrinda 2 (1.,2.)  **OK**
c                     print *,'ux(1)=',ux(1)
c                     print *,'uy(1)=',uy(1)

c      esm(1,1) =  24.d0*ux(1) + 2.d-8
c      esm(1,2) =  -12.d0*uy(1) + 1.d-8
c      esm(2,1) =  -12.d0*uy(1) + 1.d-8
c      esm(2,2) =  8.d0*uy(1) + 2.d-8
c                     print *,'esm(1,1)=',esm(1,1)
c                     print *,'esm(1,2)=',esm(1,2)
c                     print *,'esm(2,1)=',esm(2,1)
c                     print *,'esm(2,2)=',esm(2,2)
c                     pause
c***Hrinda 3 (1.,2.)  **OK**
c                     print *,'ux(1)=',ux(1)
c                     print *,'uy(1)=',uy(1)

cc      esm(1,1) =  24.d0*ux(1)*ux(1) + 2.d-10
cc      esm(1,2) =  -16.d0
cc      esm(2,1) =  -36.d0
cc      esm(2,2) =  6.d0*uy(1) + 2.d-10
c                     print *,'esm(1,1)=',esm(1,1)
c                     print *,'esm(1,2)=',esm(1,2)
c                     print *,'esm(2,1)=',esm(2,1)
c                     print *,'esm(2,2)=',esm(2,2)
c                     pause


c**Poly Test
c     print *,'iel inside elstif=',iel
c                     If(iel .eq. 1)then
c                     esm(1,1)=1.d6* .0832*ux(1)**8.
c                     esm(2,2)=1.d6*-.7298*uy(1)**7.
c                     esm(3,3)=1.d6* 2.7279*uz(1)**6.
c                     esm(4,4)=1.d6*-5.6442*ux(2)**4.
c                     esm(5,5)=1.d6* 7.0200*uy(2)**3.
c                     esm(6,6)=1.d6*-5.3187*uz(2)**2.
c                     Else
c                     esm(1,1)=1.d6* 2.3591*ux(1)**1.
c                     esm(2,2)=1.d6*-.5451*uy(1)
c                     csm(3,3)=1.d6* .0476*uz(1)
c                     esm(4,4)=0.d0
c                     esm(5,5)=0.d0
c                     esm(6,6)=0.d0
c                     Endif

c                     esm(1,1)=(1./((ux(1)-.6+(1./(ux(1)+1.d-6))))) +
c     c                   (1./((ux(1)-1.8+(.85/(ux(1)+1.d-6)))))


c                     esm(1,1)=(1./(((ux(1)+1.d-6)**2.)*(ux(1)-.6+(.1/(ux(1)+1.d-6))))))+
c     c                   (1./(((ux(1)+1.d-6)**2.)*(ux(1)-1.8+(.85/(ux(1)+1.d-6)))))
cc                    csm(1,1)=(1./(((ux(1)+1.d-6)**2.)*(ux(1)-.6+(.1/(ux(1)+1.d-6))))))+
cc    c                 (1./(((ux(1)+1.d-6)**2.)*(ux(1)-1.8+(.85/(ux(1)+1.d-6)))))
cc    c                             -6.d0


c                     esm(1,1)= (-4.*(509.d0*ux(1)-87.d0-(22.d0/(ux(1)+1.d-6))+
c     c                     300.d0*ux(1)**3.-
c     c       720.d0*ux(1)**2.)       / ((10.d0*ux(1)**2.-6.d0*ux(1)+1.d0)*
c     c       (20.d0*ux(1)**2.-36.d0*ux(1)+17.d0))) + 1.d-6

c                     print *,'esm(1,1)=',esm(1,1)
c                     print *,'esm(2,2)=',esm(2,2)
c                     print *,'esm(3,3)=',esm(3,3)
c                     print *,'csm(4,4)=',esm(4,4)
c                     print *,'csm(5,5)=',csm(5,5)
c                     print *,'csm(6,6)=',esm(6,6)
c                     print *,'esm(1,2)=',esm(1,2)
c                     print *,'csm(2,1)=',esm(2,1)
c                     print *,'csm(1,3)=',esm(1,3)
c                     pause

                      Endif

                      GOTO 1010

     If(ityp .eq. 12)then
     DO 22 I=1,6
     DO 22 J=1,6
  22 csm(I,J)=0.0
     Do 11 I=1,2
     finx(i)=0.0
     finy(i)=0.0
  11 finz(i)=0.0
     endif

c  ityp = 6 for 2-D, =12 for 3-D
c     ityp = 12
c     itruss = 1 for Green's,  = 2 for Engineering
                      itruss = 1
c                     itruss = 2
c                     itruss = 3
                      POISS = .5d0

c                      print *,'iarc inside elstif=',iarc
c                     print *,'area inside elstif=',area
c                     print *,'youngm inside elstif=',youngm
c                     print *,'maxdofpe=',maxdofpe
c                     stop
c                     print *,'ityp inside elstif=',ityp

c                     stop
                      If(iarc .eq. 1 .or. iarc .eq. 0)then
c*****************************************************************
cxxx      If(ityp.eq.12)then  ! Space truss element

c     print *,'Reading in Space Truss Stiffness In elstif'

c..... Three Dimensional Truss Element Stiffness matrix
c       a=dxc
c       c=dyc

       a=area
                       e=youngm

c     print *,'a = ',a
c     print *,'e = ',e
c     pause
c     print *,'xi  xj  yi  yj  zi  zj= ',xi,xj,yi,yj,zi,zj

c                     pause

cc     print *,'ux(1) & ux(2) = ',ux(1),ux(2)
cc     print *,'uy(1) & uy(2) = ',uy(1),uy(2)
cc     print *,'uz(1) & uz(2) = ',uz(1),uz(2)
cx                    x(1)=xi
cx                    x(2)=xj
cx                    y(1)=yi
cx                    y(2)=yj
cx                    z(1)=zi
cx                    z(2)=zj
cx     print *,'x(1)  x(2) = ',x(1),x(2)
cx     print *,'y(1)  y(2) = ',y(1),y(2)
cx     print *,'z(1)  z(2) = ',z(1),z(2)
cx                    pause
cx                     dcltax=x(2)-x(1)
cx       deltay=y(2)-y(1)
cx       deltaz=z(2)-z(1)
                       deltax=xj-xi
       deltay=yj-yi
       deltaz=zj-zi

c                     print *,'dcltay=',deltay
c                     pause

c       deltax=-x(2)+x(1)
c       deltay=-y(2)+y(1)
c       deltaz=-z(2)+z(1)

c       deltaxi=x(2)-ux(2) - ( x(1)-ux(1) )
c        deltaxi=x(2)+ux(2) - ( x(1)+ux(1) )
cx         deltaxi=x(2)-x(1) + ux(2)-ux(1)
        deltaxi=xj-xi + ux(2)-ux(1)
cc     print *,'A check'
c        deltayi=y(2)-uy(2) - ( y(1)-uy(1) )
c        deltayi=y(2)+uy(2) - ( y(1)+uy(1) )
cx         deltayi=y(2)-y(1) + uy(2)-uy(1)
       deltayi=yj-yi + uy(2)-uy(1)
c        print *,'deltayi=',deltayi
cc        print *,'y(2) y(1) = ',y(2),' ',y(1)
cc        print *,'uy(2) uy(1) = ',uy(2),' ',uy(1)

c       deltazi=z(2)-uz(2) - ( z(1)-uz(1) )
c       deltazi=z(2)+uz(2) - ( z(1)+uz(1) )
cx         deltazi=z(2)-z(1) + uz(2)-uz(1)
       deltazi=zj-zi + uz(2)-uz(1)

c       print *,'deltax=',deltax
c       print *,'deltay=',deltay
c       print *,'deltaz=',deltaz
e       print *,'deltaxi=',deltaxi
c       print *,'deltayi=',deltayi
c       print *,'deltazi=',deltazi


c                       pause
c     print *,'B check'
```

```
c                   pause
      xl=dsqrt(deltax**2.d0 + deltay**2.d0 + deltaz**2.d0)
      xli=dsqrt(deltaxi**2.d0 + deltayi**2.d0 + deltazi**2.d0)

c                   print *,'xl =',xl
c                   print *,'xli=',xli

      cx=deltax/xl
      cy=deltay/xl
      cz=deltaz/xl
      const=e*a/xl

      AN = 0.0
      ALAM = xl/xli
      X21D = deltaxi
      Y21D = deltayi
      Z21D = deltazi

      C(1) = -X21D
      C(2) = -Y21D
      C(3) = -Z21D
      C(4) = -C(1)
      C(5) = -C(2)
      C(6) = -C(3)

      P(1)=ux(1)
      P(2)=uy(1)
      P(3)=uz(1)
      P(4)=ux(2)
      P(5)=uy(2)
      P(6)=uz(2)
      dx= ux(2)-ux(1)
      dy= uy(2)-uy(1)
      dz= uz(2)-uz(1)

      ALO2=xl*xl
C..Glenn's modify
      B1(1) = -deltax/ALO2
      B1(2) = -deltay/ALO2
      B1(3) = -deltaz/ALO2
      B1(4) = -B1(1)
      B1(5) = -B1(2)
      B1(6) = -B1(3)

      EGR=0.D0
      Do 100 I=1,6
 100  EGR=EGR+B1(I)*P(I)

      EGR=EGR+.5*(dx*dx+dy*dy+dz*dz)/ALO2

      EST=EGR
c                   print *,'EST=',est
c                   stop
      If (itruss .eq. 2)then
                    Print *,'Using Engineering Strain Element!'
c                   STOP
c_____________________________________
      EST=2.*xl*EGR/(xli+xl)
c_____________________________________
      Endif

      If (itruss .eq. 3)then
                    Print *,'Using Log-Strain Element!'
c                   STOP
c_____________________________________
      EST=LOG(xli/xl)
                    print *,'EST=',est
c                   pause
c_____________________________________
      Endif


cc     print *,'EST = ',est
c** "AN" is the internal force from Crisfield, vol I, p.88
      AN = E*A*EST
c                   print *,'E=',E
c                   print *,'A=',A
c     print *,'AN = ',AN
c       print *,'xl = ',xl
c       print *,'xli = ',xli
c                   pause
c                    stop

c     If(1first .eq. 1)then
c     write(12,*)'Internal Force AN'
c     endif
c     write(12,51) AN
c 51   format(e12.4)
c     print *,'AN2 = ',AN2
c**************************************
c     Endif

      EA = E*A
c       AN2 = A*E*(xl-xli)/xl

c                   stop
c     print *,'AN2 = ',AN2
c     pause
cc      CON1 = 1.0D0/xlorig**3
      CON1 = 1.0D0/xl**3
c** EA is very large compared to AN*ALAM
c                   print *,'itruss inside elstif = ',itruss

c                   print *,'lm3a in elstif=',(lm(ic),ic=1,6)
c                   pause

c                   print *,'itruss=',itruss

      If(itruss .eq. 1)then
      CON1 = E*A*CON1
c                   print *,'E=',E
c                   print *,'A=',A
c                   print *,'CON1 = ',CON1
c                   pause
      elseif(itruss .eq. 2)then
c                   ALAM=xl/AN
      CON1 = ALAM*ALAM*CON1*(EA-AN*ALAM)
      endif

                    If(itruss .eq. 3) then
                    CON1=CON1*ALAM**4*(EA-(1.d0+.5d0*POISS)*AN)
                    endif

c                   print *,'lm3b in elstif=',(lm(ic),ic=1,6)
c                   pause

c     print *,'CON1 & CON2 = ',con1,'  ',con2
cc     pause
      DO 33 J=1,6
      DO 33 I=1,J
c                   print *,'I=',i,' J=',j
c     print *,'C(I) & C(J) & CON1 = ',c(i),c(j),con1

c                   print *,'lm3bc in elstif=',lm(J)
c                   pause

   33 csm(I,J)=CON1*C(I)*C(J)

c                   print *,'lm3c in elstif=',(lm(icc),icc=1,6)
c                   pause

cvv     If(itruss .ge. 1)then
      If(itruss .eq. 1)then
      CON2=AN/xl
c                   print *,'CON2 =',con2
c                   stop
cvv     Elseif(itruss .eq. 2)then
      Else
      CON2=CON2*ALAM
      Endif
c                     print *,'CON2 =',con2
c                   pause
c       print *,'xl = ',xl
c       print *,'xli = ',xli

c                   print *,'lm3d in elstif=',(lm(ic),ic=1,6)
c                   pause

c                     stop

c     con2=0.
c..Crisfield original..
c     esm(1,1)=esm(1,1)+CON2
c     esm(2,2)=esm(2,2)+CON2
c     esm(3,3)=esm(3,3)+CON2
c     esm(4,4)=esm(4,4)+CON2
c     esm(5,5)=esm(5,5)+CON2
c     esm(6,6)=esm(6,6)+CON2
c     esm(1,4)=esm(1,4)-CON2
c     esm(2,5)=esm(2,5)-CON2
c     esm(3,6)=esm(3,6)-CON2

c..Duc's.....
      esm(1,1)=esm(1,1)+CON2
      esm(2,2)=esm(2,2)+CON2
      esm(3,3)=esm(3,3)+CON2
      esm(4,4)=esm(4,4)+CON2
      esm(5,5)=esm(5,5)+CON2
      esm(6,6)=esm(6,6)+CON2
      esm(1,4)=esm(1,4)-CON2
      esm(2,5)=esm(2,5)-CON2
      esm(3,6)=esm(3,6)-CON2


c                   If(istep .eq. 440)then
c                   print *,'CON2=',con2
c                   print *,'esm(1,1)=',esm(1,1)
c                   print *,'esm(2,2)=',esm(2,2)
c                   print *,'esm(3,3)=',esm(3,3)
c                   print *,'esm(4,4)=',esm(4,4)
c                   print *,'esm(5,5)=',esm(5,5)
c                   print *,'esm(6,6)=',esm(6,6)
c                   print *,'esm(1,4)=',esm(1,4)
c                   print *,'esm(2,5)=',esm(2,5)
c                   print *,'esm(3,6)=',esm(3,6)
c                   pause
c                   endif
c                     print *,'xl = ',xl
c       print *,'xli = ',xli
c     stop
c     esm(1,2)=0.
c     esm(1,3)=0.
c     esm(1,5)=0.
c     esm(1,6)=0.
c     esm(2,3)=0.
c     esm(2,4)=0.
c     esm(2,6)=0.
c     esm(3,4)=0.
c     esm(3,5)=0.

c..Crisfield modified
c     esm(1,1)=esm(1,1)+CON2
c     esm(2,2)=esm(2,2)+CON2
c     esm(3,3)=esm(3,3)+CON2
c     esm(4,4)=esm(4,4)+CON2
c     esm(5,5)=esm(5,5)+CON2
c     esm(6,6)=esm(6,6)+CON2
c     esm(1,2)=esm(1,2)-CON2
c     esm(1,3)=esm(1,3)-CON2
c     esm(1,4)=esm(1,4)-CON2
c     esm(1,5)=esm(1,5)-CON2
c     esm(1,6)=esm(1,6)-CON2
c     esm(3,4)=esm(3,4)-CON2
c     esm(3,5)=esm(3,5)-CON2
```

```
c     esm(3,6)=esm(3,6)-CON2
c     esm(4,5)=esm(4,5)-CON2
c     esm(4,6)=esm(4,6)-CON2
c     esm(5,6)=esm(5,6)-CON2
c..Glenn's esm..
c     esm(1,1)=esm(1,1)+CON2
c     esm(1,2)=esm(1,2)-CON2
c     esm(2,2)=esm(2,2)+CON2
c     esm(3,3)=esm(3,3)+CON2
c     esm(3,4)=esm(3,4)-CON2
c     esm(4,4)=esm(4,4)+CON2
c     esm(4,5)=esm(4,5)-CON2
c     esm(5,5)=esm(5,5)+CON2
c     esm(5,6)=esm(5,6)-CON2
c     esm(6,6)=esm(6,6)+CON2
c       print *,'xl = ',xl
c       print *,'xli = ',xli

      DO 344 I=1,6
      DO 344 J=1,6
c                   print *,'I',i,'J',j,'esm(I,J) inside elstif=',esm(i,j)
344   esm(J,I)=esm(I,J)

c                   print *,'lm4 in elstif=',(lm(ic),ic=1,6)
c                   pause

c                   pause

c       print *,'xl = ',xl
c       print *,'xli = ',xli
c       stop

c34     print *,'csm inside elstmf= ',i,' ',j,' ',esm(i,j)
c     pause

c     print *,'xli inside elstmf = ',xli
c     print *,'deltaxi inside elstmf = ',deltaxi
c       print *,'a & e inside elstmf=',a,e
c       print *,'xl inside elstmf=',xl

C                   print *,'itruss=',itruss
C                   PAUSE
c                   stop
c       print *,'xl = ',xl
c       print *,'xli = ',xli
c                   print *,'AN=',AN
c                   print *,'X21D=',x21d
c     stop

      If(itruss .eq. 1)Then
      Finx(1) = -AN * (X21D/xl)
      Finx(2) = AN * (X21D/xl)
      Finy(1) = -AN * (Y21D/xl)
      Finy(2) = AN * (Y21D/xl)
      FinZ(1) = -AN * (Z21D/xl)
      FinZ(2) = AN * (Z21D/xl)
cvv       Elseif(itruss .gt. 1)Then
cvv       Elseif(itruss .eq. 2)Then
      Else
      Finx(1) = -AN * (X21D/xli) * ALAM
      Finx(2) = AN * (X21D/xli) * ALAM
      Finy(1) = -AN * (Y21D/xli) * ALAM
      Finy(2) = AN * (Y21D/xli) * ALAM
      FinZ(1) = -AN * (Z21D/xli) * ALAM
      FinZ(2) = AN * (Z21D/xli) * ALAM
      Endif
c**Internal force for Function Tests
1010                Continue !dummy
c     AE=5.e7
c     xl=2500.125
c     fintk(1) = (AE/xl**3.)*(625.d0*u1+37.5d0*u1**2.+
c     $         (u1**3.)/2.)
C***Arora's nonlinear 1
c     fintk(1) = (3.d0*u1+10.d0)*u1 + 12.d0*u2*u2
c     fintk(2) = 24.d0*u1*u2 + 4.d0*u2

C***Arora's nonlinear 2 sol=(-.33, -.5)  **OK**
cOK       finx(1) = 24.d0*ux(1) - 12.d0*uy(1)
cOK       finy(1) = -12.d0*ux(1) + 8.d0*uy(1)

cOK       print *,'Finx(1) inside elstmf = ',finx(1)
cOK       print *,'Finy(1) inside elstmf = ',finy(1)
c                   pause

C***J. Dennis's opt. book page 172
c     fintk(1) = u1 + u2
c     fintk(2) = u1*u1 + u2*u2

C***Arora's nonlinear soln=(0,0)  **OK**
      finx(1) = 10.d0*ux(1) + 2.d0*uy(1)
      finy(1) = 2.d0*ux(1) + 2.d0*uy(1)

C***J. Dennis opt soln=(0.00228,0.00228)
c       finx(1)=(4.d0*ux(1)*ux(1)+1.d-10)*ux(1) +
c     c         (4.d0*ux(1)*uy(1)+2.d-10)*uy(1)
c       finx(2)=(4.d0*uy(1)*ux(1)+3.d-10)*ux(1) +
c     c                   (4.d0*uy(1)*uy(1)+4.d-10)*uy(1)

c       finx(1)=(4.d0*(ux(1)**2.)+2.d-6)*ux(1) +
c     c         (4.d0*ux(1)*(uy(1))+1.d-6)*uy(1)
c       finx(2)=(4.d0*uy(1)*ux(1)+1.d-6)*ux(1) +
c     c                   (4.d0*(uy(1)**2.)+2.d-6)*uy(1)

c       csm(1,1) = 4.d0*ux(1)*ux(1) + 1.
c       csm(1,2) = 4.d0*ux(1)*uy(1) + 2.
c       esm(2,1) = 4.d0*ux(1)*uy(1) + 3.
c       esm(2,2) = 4.d0*uy(1)*uy(1) + 4.

c***Hrinda 1   **OK**
cc        finx(1)=(24.d0*ux(1)+2.d-6)*ux(1)+(-12.d0+1.d-6)*uy(1)
cc        finy(1)=(-12.d0+1.d-6)*ux(1)+(8.d0*uy(1)+2.d-6)*uy(1)

c***Hrinda 2   **OK**
c       finx(1)=(24.d0*ux(1)+2.d-8)*ux(1)+(-12.d0*uy(1)+1.d-8)*uy(1)
c       finy(1)=(-12.d0*uy(1)+1.d-8)*ux(1)+(8.d0*uy(1)+2.d-8)*uy(1)
c                   print *,'finx(1)=',finx(1)
c                   print *,'finy(1)=',finy(1)
c                   pause
c***Hrinda 3   **OK**
cc       finx(1)=(24.d0*ux(1)*ux(1)+2.d-10)*ux(1)+
cc     c         (-16.d0)*uy(1)

cc       finy(1)=(-36.d0)*ux(1)+
cc     c         (6.d0*uy(1)+2.d-10)*uy(1)

c                   print *,'finx(1)=',finx(1)
c                   print *,'finy(1)=',finy(1)

c     print *,'Finx(1) inside elstmf = ',finx(1)
c     print *,'Finy(1) inside elstmf = ',finy(1)

c**Poly Test

c                   If(iel .eq. 1)then
c       finx(1)=(1.d6*  .0832 *ux(1)**9.)
c       finy(1)=(1.d6* -.7298 *uy(1)**8.)
c       finz(1)=(1.d6* 2.7279 *uz(1)**7.)
c       finx(2)=(1.d6*-5.6442 *ux(2)**6.)
c       finy(2)=(1.d6* 7.0200 *uy(2)**5.)
c       finz(2)=(1.d6* -5.3187*uz(2)**4.)
c                   ELSE
c       finx(1)=(1.d6*  2.3591 *ux(1)**3.)
c       finy(1)=(1.d6* -.5451 *uy(1)**2.)
c       finz(1)=(1.d6*  .0476 *uz(1)**1.)
c       finx(2)=0.d0
c       finy(2)=0.d0
c       finz(2)=0.d0
c                   Endif
cc                  finx(1)=(1./(ux(1)-.6+(1./ux(1))) +
cc     c     1./(ux(1)-1.8+(.85/ux(1))))*ux(1)
c**Matlab Example
c                   finx(1)=((1./(ux(1)-.6+(1./ux(1)+1.d-6))) +
c     c                (1./(ux(1)-1.8+(.85/ux(1)+1.d-6))))*ux(1)
c                   finx(1)=((1./((ux(1)-.6+(1./ux(1)+1.d-6)))) +
c     c                (1./((ux(1)-1.8+(.85/ux(1)+1.d-6))))*ux(1)
c                   finx(1)=((1./((ux(1)-.6+(1./ux(1)+1.d-6))))) +
c     c                (1./((ux(1)-1.8+(.85/(ux(1)+1.d-6)))))*ux(1)

c                   finx(1)=((1./(ux(1)**2.)*(ux(1)-.6+(1./(ux(1)+1.d-6)))) +
c     c                (1./(ux(1)**2.)*(ux(1)-1.8+(.85/(ux(1)+1.d-6))))*ux(1)

c                   finx(1)=((1./(ux(1)**2.)*(ux(1)-.6+(.1/(ux(1)+1.d-6))))) +
c     c                (1./((ux(1)**2.)*(ux(1)-1.8+(.85/(ux(1)+1.d-6))))*ux(1)

c                   finx(1)=((1./(((ux(1)+1.d-6)**2.)*(ux(1)-.6+(.1/(ux(1)+1.d-6))))))+
c     c (1./(((ux(1)+1.d-6)**2.)*(ux(1)-1.8+(.85/(ux(1)+1.d-6))))))*
c     c (ux(1))

c                   finx(1)=((1./(((ux(1)+1.d-6)**2.)*(ux(1)-.6+(.1/(ux(1)+1.d-6))))))+
c     c (1./(((ux(1)+1.d-6)**2.)*(ux(1)-1.8+(.85/(ux(1)+1.d-6))))))*
c     c (ux(1))-(6.*ux(1))

cc                  finx(1)=((1./(((ux(1)+1.d-6)**2.)*(ux(1)-.6+(.1/(ux(1)+1.d-6))))))+
cc      c                (1./(((ux(1)+1.d-6)**2.)*(ux(1)-1.8+(.85/(ux(1)+1.d-6)))))
cc      c                         -6.d0)*ux(1)

c                   print *,'ux(1)=',ux(1)
c                   print *,'finx(1)=',finx(1)
c                   pause

c                   finx(1)= ((-4.*ux(1))*(509.d0*ux(1)-87.d0-(22.d0/(ux(1)+1.d-6))+
c     c       300.d0*ux(1)**3.-720.d0*ux(1)**2.)      /
c     c       (10.d0*ux(1)**2.-6.d0*ux(1)+1.d0)*
c     c       (20.d0*ux(1)**2.-36.d0*ux(1)+17.d0)) + (1.d-6*ux(1))

c                   esm(1,1)= -4.*(509.d0*ux(1)-87.d0-(22.d0/(ux(1)+1.d-6))+
c     c                 300.d0*ux(1)**3.-
c     c       720.d0*ux(1)**2.)       / ((10.d0*ux(1)**2.-6.d0*ux(1)+1.d0)*
c     c       (20.d0*ux(1)**2.-36.d0*ux(1)+17.d0)) + 1.d-6

c                   pause


c                   pause
c       Finy(1) = 1.d-6
c       Finy(2) = 1.d-6
c       FinZ(1) = 1.d-6
c       FinZ(2) = 1.d-6


c...deltaxi=X21D...AN = E*A*EST...
c       Finx(1) = -((xl-xli)*a*e/xl) * (deltaxi/xli)
c       Finx(2) = ((xl-xli)*a*e/xl) * (deltaxi/xli)
cc      Finx(1) = ((xl-xli)*a*e/xl) * (deltaxi/xli)
cc      Finx(2) = -((xl-xli)*a*e/xl) * (deltaxi/xli)
cc      print *,'AN axial load = ',AN
c       print *,'Finx(1) inside elstmf = ',finx(1)
c       print *,'Finx(2) inside elstmf = ',finx(2)
c                   pause
c       print *,'Finy(1) inside elstmf = ',finy(1)
c       print *,'Finy(2) inside elstmf = ',finy(2)
c       print *,'Finz(1) inside elstmf = ',finz(1)
c       print *,'Finz(2) inside elstmf = ',finz(2)
c       pause

c       Finy(1) = -AN * (Y21D/xl)
c       Finy(2) = AN * (Y21D/xl)
c       Finy(1) = -AN * (Y21D/xli)
c       Finy(2) = AN * (Y21D/xli)
cc      print *,'Finy(1) = -AN * (Y21D/xl)',finy(1)
cc      print *,'Finy(2) = AN * (Y21D/xl)',finy(2)

c       Finy(1) = -((xl-xli)*a*e/xl) * (deltayi/xli)
c       Finy(2) = ((xl-xli)*a*e/xl) * (deltayi/xli)
```

```
cc      Finy(1) = ((xl-xli)*a*e/xl) * (deltayi/xli)
cc      Finy(2) = -((xl-xli)*a*e/xl) * (deltayi/xli)

cc     print *,'xl = ',xl
cc     print *,'xli = ',xli
cc     print *,'deltayi = ',deltayi


cc     print *,'Finy(1) inside elstmf = ',finy(1)
cc     print *,'Finy(2) inside elstmf = ',finy(2)
c     pause

c      FinZ(1) = -AN * (Z21D/xl)
c      FinZ(2) = AN * (Z21D/xl)
c      FinZ(1) = -AN * (Z21D/xli)
c      FinZ(2) = AN * (Z21D/xli)
c      FinZ(1) = -((xl-xli)*a*e/xl) * (deltaZi/xli)
c      FinZ(2) = ((xl-xli)*a*e/xl) * (deltaZi/xli)
cc      FinZ(1) = ((xl-xli)*a*e/xl) * (deltaZi/xli)
cc      FinZ(2) = -((xl-xli)*a*e/xl) * (deltaZi/xli)

cc     print *,'FinZ(1) inside elstmf = ',finZ(1)
cc     print *,'FinZ(2) inside elstmf = ',finZ(2)
cc     pause

c     print *,'esm = ',esm
c     print *,'Finx & Finy & Finz inside elstmf = ',Finx,Finy,Finz
cc     print *,'deltayi = ',deltayi
cc     print *,'springK = ',springK
c     print *,'Inside elsmf at end'
c     pause
c       stop
      endif

c                   print *,'lm5 at end elstif=',(lm(ic),ic=1,6)
c                   pause

cxxx     Endif
c-------------------------------------------------------------
c      K=126
c      J=0
c      Do 101 I=1,126
c      K=J+K
c      J=126-I
c101   Continue
c      print *,'K=',K
c      stop

chk                 If(iel .eq. 4 .and. xli .le. 13.d0)then

                    icheck=1

chk                 print *,'iel =',iel
chk                 print *,'xli =',xli

c                   print *,'esm(1,1)=',esm(1,1)
c                   print *,'esm(2,2)=',esm(2,2)
c                   print *,'esm(3,3)=',esm(3,3)
c                   print *,'esm(4,4)=',esm(4,4)
c                   print *,'esm(5,5)=',esm(5,5)
c                   print *,'esm(6,6)=',esm(6,6)
c                   print *,'esm(1,4)=',esm(1,4)
c                   print *,'esm(2,5)=',esm(2,5)
c                   print *,'esm(3,6)=',esm(3,6)
c                   pause
chk                 endif
c                   pause

c************
c
c  output of the element matrices
c
c************
c************
  7  if(iplvl.eq.0) return
cgah     write(io,8) kk
cgah  8  format(/10x,7helement,i4/10x,12hforce vector,10x,
cgah     +16hstiffness matrix)
cgah     do 9 i=1,ndofpe
cgah  9  write(io,10) ef(i),(esm(i,j),j=1,ndofpe)
c  9  write(io,10) ef(i),(esm(i,j),j=1,kl)
cgah  10  format(8x,e8.2,3x,6e8.2)
      return
c*************
c*****************
c  diagnostic output
c******************
c******************
  5  continue


c  5  write(io,6) kk
c  6  format(//10x,19hthe area of element, i4,
c     +20h is less than 0.0001/
c     +10x,39hthe node numbers are in the wrong order/
c     +10x,33hor the nodes form a straight line/
c     +10x,20hexecution terminated)
      Ifirst=Ifirst+1 !counter for writing out internal forces
      stop
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
       subroutine info_elstif(iel,xlength,xl,cx,cy,cz,xx,yy,zz,ie,je
     $,propmat,propsect,youngm,area,rho,
     $ deltaxi,deltayi,deltazi,xli,
     $ xi,xj,yi,yj,zi,zj)

       implicit real*8(a-h,o-z)
       dimension xx(500),yy(500),zz(500),ie(1),je(1),
     $         propmat(1),propsect(1)
c       dimension ux(2),uy(2)
      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/disp/ux(2),uy(2),uz(2),P(6),B1(6)
c     common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)


c
c......purposes: obtain all necessary information before calculating
c            2-d truss element stiffness matrix (in global coordinate axis)
c
c                   print *,'Chk start info_elstif'
c                   print *,'xx start info_elstif=',(xx(i),i=1,4)
c                   print *,'yy start info_elstif=',(yy(i),i=1,4)
c                   print *,'zz start info_elstif=',(zz(i),i=1,4)
c                   stop
c                   pause
c                   print *,'iel =',iel
c                   print *,'ie =',ie
c                   print *,'je =',je
      locbegin=ie(iel)
c                   print *,'locbegin=',locbegin
c                   pause
c                   print *,'je(locbegin+1)=',je(locbegin+1)
c                   print *,'je(locbegin+3)=',je(locbegin+3)
c Below is for 2D truss
c                    nodei=je(locbegin+1)/2
c     nodej=je(locbegin+3)/2
c Changed for 3D truss
                     nodei=je(locbegin+2)/3
      nodej=je(locbegin+5)/3

c                   print *,'nodei & nodej=',nodei,'  ',nodej
c                   pause
      xi=xx(nodei)
c                   print *,'xi=',xi
c                   print *,'xx(nodei)=',xx(nodei)
c                   pause
c                   print *,'check after xi'
c                   pause
      yi=yy(nodei)
c                   print *,'yi=',yi
c                   print *,'check after yi'
c                   pause
                     zi=zz(nodei)
c                   print *,'zi=',zi
c                   print *,'check after zi'
c                   pause
      xj=xx(nodej)
c                   print *,'xj=',xj
c                   print *,'check after xj'
c                   pause
      yj=yy(nodej)
c                   print *,'yj=',yj
c                   print *,'check after yj'
c                   pause
                     zj=zz(nodej)
c                   print *,'zj=',zj
c                   print *,'check after zj'
c                   pause
c                   print *,'xi & yi & zi =',xi,' ',yi,' ',zi
c                   print *,'xj & yj & zj =',xj,' ',yj,' ',zj
c                   stop

      youngm=propmat(1)
      area=propsect(1)
      rho=propmat(10)

c*** Skip the rest of this, it is done in 'call elstif'***
c                   print *,'inside info_elstif...Skipping to end'
c                   pause
                                   GO TO 175
c*********************************************************
       deltax=xj-xi
       deltay=yj-yi
c                   print *,'deltay inside info_elstif=',deltay
c                   pause
                      deltaz=zj-zi
       xlength=dsqrt(deltax**2+deltay**2+deltaz**2)
c                   print *,'xlength inside info=',xlength
c                   stop

                    xl=xlength
      cx=deltax/xlength
      cy=deltay/xlength
                     cz=deltaz/xlength
c                   print *,'cx inside info_elstif=',cx
c                   print *,'cy inside info_elstif=',cy
c                   print *,'cz inside info_elstif=',cz
c                   pause
c


c                   print *,'**ux(1)=',ux(1)
c                   print *,'**uy(1)=',uy(1)
c                   print *,'uz(1)=',uz(1)
c                   print *,'ux(2)=',ux(2)
c                   print *,'uy(2)=',uy(2)
c                   print *,'uz(2)=',uz(2)
                    pause

                     deltaxi=xj-ux(2)-(xi-ux(1))
                     deltayi=yj-uy(2)-(yi-uy(1))
cc                   deltayi=yj+uy(2)-(yi+uy(1))
                     deltazi=zj-uz(2)-(zi-uz(1))
                     xli=dsqrt(deltaxi**2.0+deltayi**2.0+deltazi**2.0)

c                   print *,'deltaxi inside info_elstif=',deltaxi
c                   print *,'deltayi inside info_elstif=',deltayi
c                   print *,'deltazi inside info_elstif=',deltazi
c                   print *,'xli=',xli
c                   pause

175   CONTINUE !dummy line
```

```
      return
      end

c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine info_ex(iel,xlength,xl,cx,cy,cz,xx,yy,zz,ie,je
     $,propmat,propsect,youngm,area,rho,
     $ deltaxi,deltayi,deltazi,xli,
     $ xi,xj,yi,yj,zi,zj,AE,zee,dLength)

      implicit real*8(a-h,o-z)
      dimension xx(500),yy(500),zz(500),ie(1),je(1),
     $        propmat(1),propsect(1)
c      dimension ux(2),uy(2)

      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/disp/ux(2),uy(2),uz(2),P(6),B1(6)
c      common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)


c
c......purposes: obtain all necessary information before calculating
c           2-d truss element stiffness matrix (in global coordinate axis)
c
c                    print *,'Chk start info_elstif'
c                    print *,'xx start info_elstif=',(xx(i),i=1,4)
c                    print *,'yy start info_elstif=',(yy(i),i=1,4)
c                    print *,'zz start info_elstif=',(zz(i),i=1,4)
c                    stop
c                    pause
c                    print *,'iel =',iel
c                    print *,'ie =',ie
c                    print *,'je =',je
      locbegin=ie(iel)
c                    print *,'locbegin=',locbegin
c                    pause
c                    print *,'je(locbegin+1)=',je(locbegin+1)
c                    print *,'je(locbegin+3)=',je(locbegin+3)
c Below is for 2D truss
c                     nodei=je(locbegin+1)/2
c     nodej=je(locbegin+3)/2
c Changed for 3D truss
                      nodei=je(locbegin+2)/3
      nodej=je(locbegin+5)/3

c                    print *,'nodei & nodej=',nodei,'  ',nodej
c                    pause
      xi=xx(nodei)
c                    print *,'xi=',xi
c                    print *,'xx(nodei)=',xx(nodei)
c                    pause
c                    print *,'check after xi'
c                    pause
      yi=yy(nodei)
c                    print *,'yi=',yi
c                    print *,'check after yi'
c                    pause
                      zi=zz(nodei)
c                    print *,'zi=',zi
c                    print *,'check after zi'
c                    pause
      xj=xx(nodej)
c                    print *,'xj=',xj
c                    print *,'check after xj'
c                    pause
      yj=yy(nodej)
c                    print *,'yj=',yj
c                    print *,'check after yj'
c                    pause
                      zj=zz(nodej)
c                    print *,'zj=',zj
c                    print *,'check after zj'
c                    pause
c                    print *,'xi & yi & zi =',xi,' ',yi,' ',zi
c                    print *,'xj & yj & zj =',xj,' ',yj,' ',zj
c                    stop

      youngm=propmat(1)
      area=propsect(1)
      rho=propmat(10)

c*** Skip the rest of this, it is done in 'call elstif'***
c                    print *,'inside info_elstif...Skipping to end'
c                    pause
                                   GO TO 175
c*****************************************************
       deltax=xj-xi
       deltay=yj-yi
c                    print *,'deltay inside info_elstif=',deltay
c                    pause
                       deltaz=zj-zi
       xlength=dsqrt(deltax**2+deltay**2+deltaz**2)
c                    print *,'xlength inside info=',xlength
c                    stop

                     xl=xlength
      cx=deltax/xlength
      cy=deltay/xlength
                      cz=deltaz/xlength
c                    print *,'cx inside info_elstif=',cx
c                    print *,'cy inside info_elstif=',cy
c                    print *,'cz inside info_elstif=',cz
c                    pause
c


c                    print *,'ux(1)=',ux(1)
c                    print *,'uy(1)=',uy(1)
c                    print *,'uz(1)=',uz(1)
c                    print *,'ux(2)=',ux(2)
c                    print *,'uy(2)=',uy(2)
c                    print *,'uz(2)=',uz(2)
c                    pause

                      deltaxi=xj-ux(2)-(xi-ux(1))
                      deltayi=yj-uy(2)-(yi-uy(1))
cc                    deltayi=yj+uy(2)-(yi+uy(1))
                      deltazi=zj-uz(2)-(zi-uz(1))
                      xli=dsqrt(deltaxi**2.0+deltayi**2.0+deltazi**2.0)

c                    print *,'deltaxi inside info_elstif=',deltaxi
c                    print *,'deltayi inside info_elstif=',deltayi
c                    print *,'deltazi inside info_elstif=',deltazi
c                    print *,'xli=',xli
c                    pause

175   CONTINUE !dummy line
      return
      end

c=====================================================
c     The following routines are extracted from file ducsparsesolver.f
c     see cd ~/cee/newfem/*completely*/
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
      subroutine ermormd(answer,rhs,x,disamax,locate,disasum,n,relerr)
      implicit real*8(a-h,o-z)
      dimension answer(*),rhs(*),x(*)
c......
      relerr=0.
      bnorm=0.
      disamax=0.
      disasum=0.
      do 1 i =1,n
      bnorm=bnorm+rhs(i)*rhs(i)
      diff=answer(i)-rhs(i)
      relerr=relerr+diff*diff
      absdis=dabs( x(i) )
      disasum=disasum+absdis
      if(absdis.gt.disamax) then
      disamax=absdis
      locate=i
      endif
 1    continue
      relerr=dsqrt(relerr)
      bnorm=dsqrt(bnorm)
      relerr=relerr/bnorm
      write(6,*) 'relerr,disamax,locate,disasum = '
      write(6,*) relerr,disamax,locate,disasum
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
      subroutine fbed(n,iu,ju,di,un,b,x,iopb,isupd,getsolve)
      implicit real*8(a-h,o-z)
      dimension iu(*),ju(*),di(*),un(*),b(*),x(*),isupd(*)
C
c     input:    iu,ju,un     given upper triangular matrix with
c                            unit diagonal matrix U in RR(U)U.
c               di           inverses of the diagonal elements
c                            of the diagonal matrix D.
c               b            right-hand side vector b.
c               n            order of the system, N>1.
c     output:   x            vector of unknowns x.
c                    print *,'****** Start fbed ******'
c                    pause
c                    print *,'****** Start fbed ******'
c                    pause
c                    print *,'****** Start fbed ******'
c                    pause
c                    print *,'getstiff=',getstiff
c                    print *,'getsolve=',getsolve
c                    pause
c                    If(getsolve .eq. 1)Then
c     di(12)=1./544453.3715
c                    Endif

c                    print *,'N inside fbed=',N
c                    print *,'IU(_) = ',(iu(ii),ii=1,N-1)
c                    pause
c                    print *,'JU(_) = ',(ju(ii),ii=1,N-1)
c     pause
c                    print *,'di start fbed=',(di(ii),ii=1,N)
c     pause
c                    print *,'UN(_) = ',(un(ii),ii=1,N-1)
c                    pause
c                    print *,'b(_)=',(b(ii),ii=1,N)
c                    pause

c                    write(7,*)'N inside fbed=',N
c                    write(7,*)'IU(_) = ',(iu(ii),ii=1,N-1)
c                    write(7,*)'JU(_) = ',(ju(ii),ii=1,3)
c                    write(7,*)'di=',(di(ii),ii=1,N)
c                    write(7,*)'UN(_) = ',(un(ii),ii=1,N-1)
c                    write(7,*)'b(_)=',(b(ii),ii=1,12)
c                    pause
      NM=N-1

      DO 10 I=1,N
c                    print *,'B(I) inside fbed=',B(I)
c                    write(7,*)'B(I) inside fbed=',B(I)
   10 X(I)=B(I)
c                    pause

      DO 40 K=1,NM
c                    print *,'K=',k
c                    print *,'NM=',nm
c                    write(7,*)'K=',k
c                    write(7,*)'NM=',nm
      IUA=IU(K)
      IUB=IU(K+1)-1
      XX=X(K)
c                    print *,'X(K)=',x(k)
c                    print *,'IUA=',iua
```

```
c                    print *,'IUB=',iub
c                    pause
c                    write(7,*)'X(K)=',x(k)
c                    write(7,*)'IUA=',iua
c                    write(7,*)'IUB=',iub

      IF(IUB.LT.IUA)GO TO 30

      DO 20 I=IUA,IUB
c                    print *,'I=',i
c                    pause
c                    print *,'JU(I)=',ju(i)
c                    write(7,*)'JU(I)=',ju(i)
c                    print *,'X(JU(I))=',X(JU(I))
c                    write(7,*)'X(JU(I))=',X(JU(I))
c                    print *,'UN(I)=',UN(I)
c                    write(7,*)'UN(I)=',UN(I)
20    X(JU(I))=X(JU(I))-UN(I)*XX

c                    print *,'X(JU(I))=',(x(ic),ic=1,n)

c                     print *,'JU(I)=',ju(i)
c                     print *,'X(JU(I))=',x(ju(i))
c                    write(7,*)'JU(I)=',ju(i)
c                    write(7,*)'X(JU(I))=',x(ju(i))
c                     pause
c                    print *,'XX=',xx
c                    write(7,*)'XX=',xx
c                    print *,'DI(K)=',DI(K)
c                    write (7,*)'DI(K)=',DI(K)
30    X(K)=XX*DI(K)
c     print *,'X(K)=',(X(ic),ic=1,n)

c                     print *,'XX=',xx
c                     print *,'DI(K)=',DI(K)
c                     print *,'X(K)=',x(K)
c                                   write(7,*)'XX=',xx
c                                   write(7,*)'DI(K)=',DI(K)
c                                   write(7,*)'X(K)=',x(K)
   40 CONTINUE
c                    pause

c                    print *,'X(5)=',x(5)
c                    print *,'X(N) before=',X(N)
      X(N)=X(N)*DI(N)
c                    print *,'DI(N)=',DI(N)
c                    print *,'N=',n
c                    print *,'X(N)=',x(n)
c                    write(7,*)'X(N)=',x(n)
c                    pause
      K=NM
   50 IUA=IU(K)
      IUB=IU(K+1)-1
      IF(IUB.LT.IUA)GO TO 70
c                    print *,'X(K)=',x(k)
c                    write(7,*)'X(K)=',x(k)
      XX=X(K)
c                    print *,'X(5)=',x(5)
      DO 60 I=IUA,IUB
c                     print *,'XX=',xx
c                     print *,'IUA=',iua
c                     print *,'IUB=',iub
c                    write(7,*)'XX=',xx
c                    write(7,*)'IUA=',iua
c                    write(7,*)'IUB=',iub
c                    pause
c                     print *,'XX=',xx
c                     print *,'UN(I)=',un(i)
c                     print *,'JU(I)=',ju(i)
c                     print *,'X(12)=',x(12)
c                     print *,'X(JU(I))=',x(ju(i))
c                    write(7,*)'XX=',xx
c                    write(7,*)'UN(I)=',un(i)
c                    write(7,*)'JU(I)=',ju(i)
c                    write(7,*)'X(12)=',x(12)
c                    write(7,*)'X(JU(I))=',x(ju(i))
c                    pause
   60 XX=XX-UN(I)*X(JU(I))
c                    print *,'XX=',xx
c                    print *,'K=',k
c                    write(7,*)'XX=',xx
c                    write(7,*)'K=',k
c                    pause
      X(K)=XX
   70 K=K-1
      IF(K.GT.0)GO TO 50
c      write(6,*) 'displacement solution= '
c      write(7,*) 'displacement solution= '
c      write(6,4) (x(i1),i1=1,n)
c                    write(*,4) (x(i2),i2=1,n)
c 4    format(5(e13.6,1x))
c                    print *,'x = ',(x(i3),i3=1,n)
c                    print *,'End of fbed'
c                                   write(7,*)'x = ',(x(i3),i3=1,n)
c                                   write(7,*)'End of fbed'
c                    stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine mvspar(diag,     kindx,coefs,rhs,istartr        ! no need array kptrs(-)
     $,n,ncoef,answer,ioper,isupern)
c************************************************************************
c     PLEASE do "NOT" distribute this source code to any
c     persons/organizations, unless obtaining the written
c     permission from Duc T. Nguyen (Tel= 757-683-3761,
c                         Email= dnguyen@odu.edu)
c************************************************************************
      implicit real*8(a-h,o-z)
c     implicit complex*16(a-h,o-z)
c......purpose: <sparse, and symmetric> matrix times vector
c......       unrolling option will soon be provided
c......       This routine has been verified to give "CORRECT" answer (June 30'00)
c     dimension diag(1),kptrs(1),kindx(1),coefs(1),rhs(1),istartr(1)
      dimension diag(1),        kindx(1),coefs(1),rhs(1),istartr(1)! no need array kptrs(-)
     $,answer(1),isupern(1)
c
c......for consecutive rows,determine if ALL nonzero(s) off-diagonals
c......are lined up in the same columns ! (for "FUTURE" unrolling option !)
c     call supern(n,kptrs,istartr,kindx,isupern)
c......initialize
      ioper=0
c
      do 3 i=1,n
 3    answer(i)=diag(i)*rhs(i)
      ioper=ioper+n
c......begin SPARSE matrix times vector
      do 1 i=1,n
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c     ii=kptrs(i)              ! no need to have kptrs(-) array
      ii=istartr(i+1)-istartr(i)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      if (ii.eq.0) go to 1
      icount=istartr(i)-1
      do 2 k=1,ii
c.......row i-th (upper triangular) operates on vector rhs(-)
      icount=icount+1
c.......find the column numbers (of the i-th row) which have non-zero
c.......off-diagonals
      jcoln=kindx(icount)
      answer(i)=answer(i)+coefs(icount)*rhs(jcoln)
      ioper=ioper+2
c.......now, column i-th (lower triangular) operates on vector rhs(-)
      answer(jcoln)=answer(jcoln)+coefs(icount)*rhs(i)
      ioper=ioper+2
 2    continue
 1    continue
      return
      end


C************************************************************************
      subroutine numfa1d(n,ia,ja,ad,an,iu,ju,di,un,ip,iup,isupd,iopf)
c      subroutine numfa1d(n,ia2,ja,ad,an,iu,ju,di,un,ip,iup,isupd,iopf)
      implicit real*8(a-h,o-z)
c     implicit complex*16(a-h,o-z)
      dimension ia(*),ja(*),ad(*),an(*),iu(*),ju(*),di(*),un(*)
c      dimension ia2(*),ja(*),ad(*),an(*),iu(*),ju(*),di(*),un(*)
      dimension ip(*),iup(*),isupd(*)
C......purpose: numerical factorization
c     input:      ia,ja,an,ad    given matrix A in RR(U)U.
c              iu,ju         structure of resulting matrix U in
c                            RR(U)O.
c              n             order of matrices A and U.
c     output:     un            numerical values of the nonzeros of
c                            matrix U in RR(U)O.
c              di            inverse of the diagonal matrix D.
c     working space:  ip        of dimension N.  Chained lists of rows
c                            associated with each column.
c                iup         of dimension N.  Auxiliary pointers to
c                            portions of rows.
c                 di         is used as the expanded accumulator.
c                   print *,'****** start of numfa1d *******'
c                   pause
c                   print *,'****** start of numfa1d *******'
c                   pause
c                   print *,'****** start of numfa1d *******'
c                   pause
c                   print *,'n start numfa1d=',n
c                   pause
      ncoef1=ia(n+1)-1
c      ncoef1=ia2(n+1)-1
c     print *, 'inside numfa1: ad(-) = ',(ad(ic),ic=1,n)
c                   pause
c     print *, 'inside numfa1: an(-) = ',(an(ic),ic=1,ncoef1)
c     print *, 'inside numfa1: an(-) = ',(an(ic),ic=1,30)
c                   pause

c                   print *,'ia2(_) inside numfa1=',(ia2(i),i=1,n)
c                   stop
c     write(6,*) 'inside numfa1: ad(-) = ',(ad(ic),ic=1,n)
c     write(6,*) 'inside numfa1: an(-) = ',(an(ic),ic=1,ncoef1)
c                   print *,'ip(_)=',(ip(ic),ic=1,9)
c                   print *,'JU(_)=',(ju(ic),ic=1,3)
c                   print *,'UN(_)@start numfa1d=',(un(ic),ic=1,n)
c                   print *,'DI(_)@start numfa1d=',(ju(ic),ic=1,n)
c                   pause
c                                  print *,'UN(1)=',un(1)
c                                  print *,'UN=',un
c     print *, 'inside numfa1: IU(-) = ',(IU(ic),ic=1,n)

c                   pause
c

      DO 10 J=1,N
   10 IP(J)=0
c......Begin of of 1-st (nested) loop: outer-most loop, for each i-th row
      DO 130 I=1,N
c                   print *,'I @ start 1st loop=',i
c                   pause
c     print *, 'outer-most loop 130 I= ',i
c     print *,'outer-most loop 130 I= ',i
      IH=I+1
      IUA=IU(I)
      IUB=IU(IH)-1
c                   print *,'IUA inside NUMFAC=',iua
c                   print *,'IUB inside NUMFAC=',iub
c                   pause
c     print *, 'NUMFAC: i,n,ih,iu(i),iua,iu(ih),iub ='
c     print *, i,n,ih,iu(i),iua,iu(ih),iub

c     write(6,*) 'NUMFAC: i,n,ih,iu(i),iua,iu(ih),iub ='
c     write(6,*) i,n,ih,iu(i),iua,iu(ih),iub
c                                  print *,'UN(1)before goto40=',un(1)
```

```
      IF(IUB.LT.IUA)GO TO 40

      DO 20 J=IUA,IUB
      DI(JU(J))=0.
c     write(6,*) 'NUMFAC,loop 20: j,iua,iub,ju(j),di(ju(j)) = '
c     write(6,*) j,iua,iub,ju(j),di(ju(j))
 20   continue
c                 print *,'di(ju(j))=',(di(ic),ic=ju(iua),ju(iub))
c                 pause

      IAA=IA(I)
      IAB=IA(IH)-1
c     IAA=IA2(I)
c     IAB=IA2(IH)-1

c     write(6,*) 'NUMFAC: i,ia2(i),iaa,ih,ia2(ih),iab= '
c     write(6,*) i,ia2(i),iaa,ih,ia2(ih),iab
      IF(IAB.LT.IAA)GO TO 40

      DO 30 J=IAA,IAB
c                                  print *,'UN(1)after 30=',un(1)
c  30 DI(JA(J))=AN(J)
c  40 DI(I)=AD(I)
      DI(JA(J))=AN(J)
c                 print *,'J inside numfa1d=',j
c                 pause
c                 print *,'AN(J) inside numfa1d=',AN(J)
c                 pause
c                 print *,'DI(JA(J)) 30 inside numfa1d=',DI(JA(J))
c                 pause
c     write(6,*) 'NUMFAC:loop30: j,iaa,iab,an(j),ja(j),di(ja(j))= '
c     write(6,*) j,iaa,iab,an(j),ja(j),di(ja(j))
 30   continue

 40   DI(I)=AD(I)
c                 print *,'DI(I) 40=',AD(I)
c                 pause
c                                  print *,'UN(1)after 40=',un(1)
c     write(6,*) 'NUMFAC: i,ad(i),di(i)= '
c     write(6,*) i,ad(i),di(i)
c                 print *,'NUMFAC: i,ad(i),di(i)=',i,ad(i),di(i)
      LAST=IP(I)
c                 print *,'LAST=',last
c                 pause

c     write(6,*) 'NUMFAC: i,ip(i),last= '
c     write(6,*) i,ip(i),last
c                                  print *,'UN(1)before goto90=',un(1)
      IF(LAST.EQ.0)GO TO 90

      LN=IP(LAST)
c                 print *,'LN=',LN

c     write(6,*) 'NUMFAC: last,ip(last),ln = '
c     write(6,*) last,ip(last),ln
c......begin of 2-nd (nested) loop: considering all APPROPRIATED previous
c......                     rows (any appropriated rows 1--->i-1)
   50 L=LN
      LN=IP(L)
      IUC=IUP(L)
      IUD=IU(L+1)-1
      UM=UN(IUC)*DI(L)
c                 print *,'IUC=',iuc
c                 print *,'IUD=',iud
c                 print *,'UM=',um

c     write(6,*) 'NUMFAC: l,ip(l),ln,iup(l),iuc = '
c     write(6,*) l,ip(l),ln,iup(l),iuc
c     write(6,*) 'NUMFAC: iu(L+1),iud,l,di(l),iuc,un(iuc),um= '
c     write(6,*) iu(L+1),iud,l,di(l),iuc,un(iuc),um
c......begin of 3-rd (nested) inner-most loop: considering all APPROPRIATED
c......                          columns (any columns i--->n)
c                 print *,'start 3rd nested loop'

c                 stop
c                 print *,'IUC=',iuc
c                 print *,'IUD=',iud
c                 print *,'UN(1)=',un(1)
      DO 60 J=IUC,IUD
c                 print *,'UN(1)after 60=',un(1)
      JJ=JU(J)
c     write(6,*) 'NUMFAC:loop60:J,di(jj)= '
c     write(6,*) J,di(jj)
c  60 DI(JJ)=DI(JJ)-UN(J)*UM
c                 print *,'DI(JJ)=',di(jj)
c                 print *,'J=',j
c                 print *,'UN(J)=',un(j)
c                 print *,'UM=',um
cccccccccccccccccccccccccccccccccccccccccccccccc
c                 UN(1)=0.0
cccccccccccccccccccccccccccccccccccccccccccccccc
c                 print *,'UN(1)a=',un(1)
      DI(JJ)=DI(JJ)-UN(J)*UM
c                 print *,'JJ=',jj
c                 print *,'DI(JJ)=',di(jj)

c                 write(6,*) 'NUMFAC:loop60:j,iuc,iud,ju(j),jj,um,un(j),di(jj)= '
c     write(6,*) j,iuc,iud,ju(j),jj,um,un(j),di(jj)
 60   continue
      UN(IUC)=UM
c                 print *,'UN(1)b=',un(1)
      IUP(L)=IUC+1
c                 print *,'UN(IUC)=',un(iuc)
c                 print *,'IUP(L)=',iup(l)

c     write(6,*) 'NUMFAC: um,iuc,iud,un(iuc),l,iup(l) = '
c     write(6,*) um,iuc,iud,un(iuc),l,iup(l)
      IF(IUC.EQ.IUD)GO TO 80
      J=JU(IUC+1)
      JJ=IP(J)
c     write(6,*) 'NUMFAC: iuc,ju(iuc+1),j,ip(j),jj= '
c     write(6,*) iuc,ju(iuc+1),j,ip(j),jj
      IF(JJ.EQ.0)GO TO 70
      IP(L)=IP(JJ)
c     write(6,*) 'NUMFAC: jj,ip(jj)= '
c     write(6,*) jj,ip(jj)
      IP(JJ)=L
c     write(6,*) 'NUMFAC: jj,ip(jj),l,ip(l)= '
c     write(6,*) jj,ip(jj),l,ip(l)
      GO TO 80
   70 IP(J)=L
      IP(L)=L
c     write(6,*) 'NUMFAC: l,j,ip(j),ip(l),last= '
c     write(6,*) l,j,ip(j),ip(l),last
c......the following go to statement is equivalent to 2-nd nested loop
c......for factorization
c......
c                 print *,'Check numfac !'
c                 print *,'L=',L,'  LAST=',last
c                 stop
   80 IF(L.NE.LAST)GO TO 50
c......
  90  continue
c     write(6,*) 'di(i)= ', di(i)
c      print *, 'di(i)= ', di(i)
c                 pause

      DI(I)=1./DI(I)
c                  print *,'I=',i
c      print *, '1./di(i) after 90 = ', di(i)
c                 pause

c     write(6,*) 'NUMFAC: i,di(i)=1/di(i),iub,iua= '
c     write(6,*) i,di(i),iub,iua
c                 print *,'IUA=',iua
c                 print *,'IUB=',iub

cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Changed Duc's code from " .LT. to .LE. "
      IF(IUB.LT.IUA)GO TO 120
c     IF(IUB.LE.IUA)GO TO 120
c     IF(IUB.GT.IUA)GO TO 120
cccccccccccccccccccccccccccccccccccccccccccccccccccccc

      DO 100 J=IUA,IUB
c 100 UN(J)=DI(JU(J))
c                                  print *,'UN(1)before chg=',un(1)
c                 print *,'JU(J)=',ju(j)
c                 print *,'DI(JU(J))=',di(ju(j))
      UN(J)=DI(JU(J))
c                                  print *,'UN(1)after chg=',un(1)
c                                  print *,'UN(1)c=',un(1)
c     write(6,*) 'NUMFAC:loop100: j,iua,iub,ju(j),di(ju(j)),un(j)= '
c     write(6,*) j,iua,iub,ju(j),di(ju(j)),un(j)
c                  print *,'I=',i
c      print *, '1./di(i) chk 1 = ', di(i)
c                 pause
 100  continue
      J=JU(IUA)
      JJ=IP(J)
c     write(6,*) 'NUMFAC: iua,ju(iua),j,ip(j),jj= '
c     write(6,*) iua,ju(iua),j,ip(j),jj
      IF(JJ.EQ.0)GO TO 110
      IP(I)=IP(JJ)
c     write(6,*) 'NUMFAC: jj,ip(jj)= '
c     write(6,*) jj,ip(jj)
      IP(JJ)=I
c     write(6,*) 'NUMFAC: jj,ip(jj),i,ip(i)= '
c     write(6,*) jj,ip(jj),i,ip(i)
      GO TO 120
  110 IP(J)=I
      IP(I)=I
c     write(6,*) 'NUMFAC: i,j,ip(j),ip(i)= '
c     write(6,*) i,j,ip(j),ip(i)
c                  print *,'I=',i
c      print *, '1./di(i) chk 2 = ', di(i)
c                 pause
  120 IUP(I)=IUA
c     write(6,*) 'NUMFAC: iua,i,iup(i)= '
c     write(6,*) iua,i,iup(i)
c                 print *,'NUMFAC: iua,i,iup(i)=',iua,i,iup(i)
c                 stop
C      print *,'check numfac 2'
c                  print *,'I=',i
c      print *, '1./di(i) chk 3 = ', di(i)
c                 pause
  130 CONTINUE

c     WRITE(6,*),'End of numfa1d'
c                 print *,'DI at end of numfa1d=',(di(iz),iz=1,n)
c                 print *,'End of numfa1d'
c                 print *,'*********************'
c                 pause
c                 print *,'*********************'
c                 pause
c                 print *,'*********************'
c                 pause
c                 print *,'*********************'
c                 pause
c                 stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
      subroutine symfactd(n,ia,ja,iu,ju,ip,ncoef2)
c      subroutine symfactd(n,ia2,ja,iu,ju,ip,ncoef2)
c      implicit real*8(a-h,o-z)
      dimension ia(*),ja(*),iu(*),ju(*),ip(*)
c      dimension ia2(*),ja(*),iu(*),ju(*),ip(*)
C......Purposes: Symbolic factorization
c     input:  ia(n+1),ja(ncoef1)   structure of given matrix A in RR(U)U.
c......       ia(n+1)             starting location of 1-st non-zero term
c             n                   order of matrix A and of matrix U.
c......       ncoef1 = ia(n+1)-1
```

```
c     output: iu(n+1),ju(ncoef2)   structure of resulting matrix U in RR(U)U.
c     working space: ip(n)         of dimension N.  Chained lists of rows
c                                  associated with each column.
c                                  The array IU is also used as the multiple
c                                  switch array.
c                    print *,'****************inside symfactd*******************'
c     print *, 'inside symfactd: ia(-) = ',(ia(i),i=1,n+1)
c     print *, 'inside symfactd: ja(-) = ',(ja(i),i=1,n+1)
c     print *, 'inside symfactd: ja(-) = ',(ja(i),i=1,3)
c                    pausc
c                     print *,'ia2 inside symfactd=',(ia2(i),i=1,n+1)
c                    stop

                     JI=0
                     JPP=0
                     MIN=0
                     IAA=0
                     IAB=0
                     JJ=0
                     JP=0
                     LAST=0
                     L=0
                     LH=0
                     IUA=0
                     IUB=0
                     NH=0
                     ncoef2=0
                     J=0
                     NM=0

      ncoef1=ia(n+1)-1
c     ncoef1=ia2(n+1)-1

c     print *, 'start symfactd: ia(-) = ',(ia(ic),ic=1,n+1)
c     print *, 'inside symfactd: ja(-) = ',(ja(i),i=1,ncoef1)
c     write(6,*) 'inside symfactd: ia2(-) = ',(ia2(i),i=1,n+1)
c     write(6,*) 'inside symfactd: ja(-) = ',(ja(i),i=1,ncoef1)
c                    pause
c
      NM=N-1
      NH=N+1
      DO 8 I=1,N+1
                     ju(i)=0
    8 ip(i)=0

      DO 10 I=1,N
      IU(I)=0
   10 IP(I)=0
      JP=1

      DO 90 I=1,NM
      JPI=JP
      JPP=N+JP-I
      MIN=NH
c                    print *,'I=',i
c                    print *,'JPI=',JPI
c                    print *,'JPP=',JPP
c                    print *,'MIN=',min
c                    pause
c                     print *,'IA2(I)=',IA2(I)
c                     print *,'IA2(I+1)=',IA2(I+1)
c                    write(6,*)'I=',i
c                    write(6,*)'JPI=',JPI
c                    write(6,*)'JPP=',JPP
c                    write(6,*)'MIN=',min
c                    pause
c                     write(6,*)'IA2(I)=',IA2(I)
c                     write(6,*)'IA2(I+1)=',IA2(I+1)

c                     stop
      IAA=IA(I)
      IAB=IA(I+1)-1
c     IAA=IA2(I)
c     IAB=IA2(I+1)-1

c                    print *,'IAA=',iaa
c                    print *,'IAB=',iab
c                    write(6,*)'IAA=',iaa
c                    write(6,*)'IAB=',iab
c                    pause
c     write(6,*) 'SYMFACT: i,nm,jpi,jp,n,jpp,min,iaa,iab = '
c     write(6,*) i,nm,jpi,jp,n,jpp,min,iaa,iab
c     print *,'SYMFACT: i,nm,jpi,jp,n,jpp,min,iaa,iab = '
c     print *, i,nm,jpi,jp,n,jpp,min,iaa,iab

      IF(IAB.LT.IAA)GO TO 30
c                     print *,'check 1 inside symfacd'
c                    print *,'JA(_) inside symfacd=',(JA(ic),ic=1,3)
c                    print *,'IAA=',IAA
c                    print *,'IAB=',IAB
      DO 20 J=IAA,IAB
c                    print *,'JA(J)=',JA(J)
      JJ=JA(J)
c                    print *,'chk JA(J) inside symfacd'
      JU(JP)=JJ
c     print *,'chk JU(JP) inside symfacd'
c      write(6,*) 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c      write(6,*) j,jj,jp,ju(jp),min
c       print *, 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c       print *, j,jj,jp,ju(jp),min
c                    pause
c                      stop

      JP=JP+1
c      write(6,*) 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c      write(6,*) j,jj,jp,ju(jp),min
c       print *, 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c       print *, j,jj,jp,ju(jp),min

c     IF(JJ.LT.MIN)MIN=JJ
      IF(JJ.LT.MIN) then
      min=jj
c      write(6,*) 'SYMFACT:insideloop20: min = '
c      write(6,*) min
c       print *, 'SYMFACT:insideloop20: min = '
c       print *, min

      endif
c  20  IU(JJ)=I
c                    print *,'I=',i
c                    print *,'JJ=',JJ
      IU(JJ)=I
c     print *, 'SYMFACT:insideloop20: jj,iu(jj),i = '
c     print *, jj,iu(jj),i
c     write(6,*) 'SYMFACT:insideloop20: jj,iu(jj),i = '
c     write(6,*) jj,iu(jj),i
   20 continue

c                     print *,'chk 1a inside sysfactd'
c                    write(6,*)'chk 1a inside sysfactd'
c                     stop
   30 LAST=IP(I)
c     write(6,*) 'SYMFACT: i,ip(i),last = '
c     write(6,*) i,ip(i),last
      IF(LAST.EQ.0)GO TO 60
      L=LAST
c                    print *,'LAST=',last
c                    write(6,*)'LAST=',last
c                    pause
   40 L=IP(L)
c                    print *,'chk 1b inside sysfactd'
c                    print *,'L=',L
c                    print *,'IP(L)=',IP(L)
c                    print *,'IU(L)=',IU(L)
c                    write(6,*)'chk 1b inside sysfactd'
c                    write(6,*)'L=',L
c                    write(6,*)'IP(L)=',IP(L)
c                    write(6,*)'IU(L)=',IU(L)
c                    pause
      LH=L+1
      IUA=IU(L)
      IUB=IU(LH)-1
c                    print *,'IU(LH)=',IU(LH)
c                                  print *,'chk 1c inside sysfactd'
c                    write(6,*)'IU(LH)=',IU(LH)
c                    write(6,*)'chk 1c inside sysfactd'

c     write(6,*) 'SYMFACT: last,l,ip(l),lh,iu(l),iua,iu(lh),iub = '
c     write(6,*) last,l,ip(l),lh,iu(l),iua,iu(lh),iub
c     IF(LH.EQ.I)IUB=JPI-1
      IF(LH.EQ.I) then
c                                  print *,'chk 1d inside sysfactd'
      iub=jpi-1
c          write(6,*)'chk 1d inside sysfactd'
c     write(6,*) 'SYMFACT: lh,i,jpi,iub = '
c     write(6,*) lh,i,jpi,iub
      endif
c                    print *,'chk 1e inside sysfactd'
c                    write(6,*)'chk 1e inside sysfactd'

      IU(I)=I
c                    print *,'IU(I)=',IU(I)

c     write(6,*) 'SYMFACT: i,iu(i) = '
c     write(6,*) i,iu(i)
      DO 50 J=IUA,IUB
      JJ=JU(J)
c                    print *,'JJ=',jj
c     write(6,*) 'SYMFACT:insideloop50: j,iua,iub,ju(j),jj,iu(jj),i = '
c     write(6,*) j,iua,iub,ju(j),jj,iu(jj),i
      IF(JU(JJ).EQ.I)GO TO 50
      JU(JP)=JJ
c     write(6,*) 'SYMFACT:insideloop50: jp,ju(jp)= '
c     write(6,*) jp,ju(jp)
      JP=JP+1
      IU(JJ)=I
c                    print *,'IU(JJ)=',IU(JJ)
c     write(6,*) 'SYMFACT:insideloop50: jp,jj,iu(jj),i= '
c     write(6,*) jp,jj,iu(jj),i
c     IF(JJ.LT.MIN)MIN=JJ
c                    print *,'chk 2 inside sysfactd'
c                    write(6,*)'chk 2 inside sysfactd'
      IF(JJ.LT.MIN) then
c     write(6,*) 'SYMFACT:insideloop50: jj,min = ',jj,min
      min=jj
c     write(6,*) 'SYMFACT:insideloop50: jj,min = ',jj,min
      endif
   50 CONTINUE
c     write(6,*) 'SYMFACT: jp,jpp,l,last,min,nh = '
c     write(6,*) jp,jpp,l,last,min,nh
c     print *, 'SYMFACT: jp,jpp,l,last,min,nh = '
c     print *, jp,jpp,l,last,min,nh
      IF(JP.EQ.JPP)GO TO 70
      IF(L.NE.LAST)GO TO 40
   60 IF(MIN.EQ.NH)GO TO 90
   70 L=IP(MIN)
c                    print *,'chk 3 inside sysfactd'
c                    write(6,*)'chk 3 inside sysfactd'
c                    stop
c     write(6,*) 'SYMFACT: min,ip(min),l = '
c     write(6,*) min,ip(min),l
      IF(L.EQ.0)GO TO 80
      IP(I)=IP(L)
c                    print *,'IP(I)=',IP(I)
c     write(6,*) 'SYMFACT: I,ip(I),i,ip(i) = '
c     write(6,*) I,ip(I),i,ip(i)
      IP(L)=I
c     write(6,*) 'SYMFACT: I,ip(I),i,ip(i) = '
c     write(6,*) I,ip(I),i,ip(i)
      GO TO 90
   80 IP(MIN)=I
c                    print *,'IP(MIN)=',IP(MIN)
c                    print *,'chk 4 inside sysfactd'
      IP(I)=I
```

```
c     write(6,*) 'SYMFACT: min,ip(min),i,ip(i) = '
c     write(6,*) min,ip(min),i,ip(i)
   90 IU(I)=JPI
c                    print *,'IU(I)=',IU(I)
      IU(N)=JP
      IU(NH)=JP
c                    print *,'IU(N)=',IU(N)
c                    print *,'IU(NH)=',IU(NH)
c     write(6,*) 'SYMFACT: i,iu(i),jpi,n,iu(n),jp,nh,iu(nh) = '
c     write(6,*) i,iu(i),jpi,n,iu(n),jp,nh,iu(nh)
c                    print *,'chk 5 inside sysfactd'
c                    write(6,*)'chk 5 inside sysfactd'
c                    stop
      ncoef2=iu(n+1)-1
c      print *,'iu(n+1) = ',iu(n+1)
c                    print *,'ncoef2 inside symfact =',ncoef2
c                    write(6,*)'ncoef2=',ncoef2
c                    stop
c      write(6,*) 'inside symfact: ncoef2= ', ncoef2
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine transad(n,m,ia,ja,iat,jat)
c      implicit real*8(a-h,o-z)
      dimension ia(*),ja(*),iat(*),jat(*)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
c......PLEASE direct your questions to Dr. Nguyen (nguyen@cee.odu.edu)
c......purposes: After symbolic factorization,ja is just a merge lists
c......       WITHOUT ordering (i.e. the nonzero column numbers of a
c......       particular row are 12, 27, 14, 46, 22, 133). Upon
c......       completion, this routine will rearrange the nonzero column
c......       numbers WITH ordering, to be ready for numerical factorization
c......       phase (i.e. 12, 14, 22, 27, 46, 133)
c     input:  ia,ja,an     given matrix in RR(C)U.
c          n           number of rows of the matrix.
c          m           number of columns of the matrix.
c     output: iat,jat,ant  transposed matrix in RR(C)O.
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
c     dimension ia(6), ja(7), iat(6), jat(7)
c     dimension ia(neq+1), ja(ncoef1), iat(neq+1), jat(ncoef1)
c     neq=5
c     ncoef1=7
c     nrows=5
c     ncols=5
c......      x  2  0  4  5   ---> input unordered col # 2,5,4
c......         x  3  4  0   ---> input unordered col # 4,3
c......            x  0  5   ---> input col # 5
c......               x  5   ---> input col # 5
c......                  x
c     ia(1)=1
c     ia(2)=4
c     ia(3)=6
c     ia(4)=7
c     ia(5)=8
c     ia(6)=8
c
c     ja(1)=2
c     ja(2)=5
c     ja(3)=4
c     ja(4)=4
c     ja(5)=3
c     ja(6)=5
c     ja(7)=5
c
c     call transad(nrows,ncols,ia,ja,iat,jat)
c     write(6,*) 'after reordered, ia(-)= ',(ia(i),i=1,neq+1)
c     write(6,*) 'after reordered, ja(-)= ',(ja(i),i=1,ncoef1)
c     stop
c     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c
c      print *,'ja at start of transd = ',(ja(i),i=1,15)
      MH=M+1
      NH=N+1
      DO 10 I=2,MH
   10 IAT(I)=0
      IAB=IA(NH)-1
      DO 20 I=1,IAB
      J=JA(I)+2
      IF(J.LE.MH)IAT(J)=IAT(J)+1
   20 CONTINUE
      IAT(1)=1
      IAT(2)=1
      IF(M.EQ.1)GO TO 40
      DO 30 I=3,MH
   30 IAT(I)=IAT(I)+IAT(I-1)
   40 DO 60 I=1,N
      IAA=IA(I)
      IAB=IA(I+1)-1
      IF(IAB.LT.IAA)GO TO 60
      DO 50 JP=IAA,IAB
      J=JA(JP)+1
      K=IAT(J)
      JAT(K)=I
c      ANT(K)=AN(JP)
   50 IAT(J)=K+1
   60 CONTINUE
c      print *,'ja at end of transd = ',(ja(i),i=1,15)
c                    pause
      call transa2d(n,m,iat,jat,ia,ja)
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c=====================================================================
=======
c     The following routines are extracted from file part2.f
c     see cd ~/cee/newfem/*completely*/
```

```
c=====================================================================

c     Table 8.7  Transposing a Sparse Matrix

c@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@
c..... file = part2.o  (only object codes) ..............
      subroutine transa2d(n,m,ia,ja,iat,jat)
      dimension ia(*),ja(*),iat(*),jat(*)
c                     dimension ja2(20)
c
c                    print *,'ia(_) inside transa2d=',(ia(i),i=1,15)
c                    print *,'ja(_) inside transa2d=',(ja(i),i=1,15)
c                    pause

      MH=M+1
      NH=N+1
      DO 10 I=2,MH
   10 IAT(I)=0

      IAB=IA(NH)-1

      DO 20 I=1,IAB
      J=JA(I)+2
      IF(J.LE.MH)IAT(J)=IAT(J)+1
   20 CONTINUE

      IAT(1)=1
      IAT(2)=1
      IF(M.EQ.1)GO TO 40
      DO 30 I=3,MH
   30 IAT(I)=IAT(I)+IAT(I-1)

c                    Do 31 I=1,20
c   31               ja2(i)=ja(i)

c                    print *,'ja(_) inside transa2d mid=',(ja(i),i=1,10)
c                    pause

   40 DO 60 I=1,N
      IAA=IA(I)
      IAB=IA(I+1)-1
      IF(IAB.LT.IAA)GO TO 60

      DO 50 JP=IAA,IAB
c                    print *,'JA(JP)=',JA(JP),'(JP)',jp
      J=JA(JP)+1
      K=IAT(J)
      JAT(K)=I
c      ANT(K)=AN(JP)
   50 IAT(J)=K+1
c                    print *,'JA(JP) 2=',JA(JP),'(JP)',jp
   60 CONTINUE
c                    print *,'JA(JP) 3=',(JA(k),k=1,20)
c
      nz=iat(m+1)-1
c      write(6,*) 'inside transa2: iet(-) = ',(iat(i),i=1,m+1)
c      write(6,*) 'inside transa2: jet(-) = ',(jat(i),i=1,nz)
c      print *, 'inside transa2 at end: iet(-) = ',(iat(i),i=1,m+1)
c      print *, 'inside transa2 at end: jet(-) = ',(jat(i),i=1,nz)
c                    stop
c                    Do 61 i=1,20
c   61               ja(i)=ja2(i)

c                    print *,'ja(_) inside transa2d at end=',(ja(i),i=1,10)
c                    pause

      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

c     Table 8.2  Complete FORTRAN Code for Sparse, Symmetrical
c             Symbolic Assembly Process

      subroutine symbass(ie,je,iet,jet,ndof,ia,ja,ncoef1)
c      subroutine symbass(ie,je,iet,jet,ndof,ia1,ia2,ja,ncoef1)
      implicit real*8(a-h,o-z)
      dimension ie(*),je(*),iet(*),jet(*),ja(*),ia(*)
c      dimension ie(*),je(*),iet(*),jet(*),ja(*),ia1(*),ia2(*)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c......PLEASE direct your questions to Dr. Nguyen (nguyen@cee.odu.edu)
c......Purposes: symmetrical, sparse symbolic assembly
c......        This code is stored under file name *symb*.f, in sub-directory
c......        cd ~/cee/newfem/complete*/part2.f
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c......Input: ie(nel+1)=locations (or pointers) of the first non-zero
c......     of each row (of element-dof connectivity info.)
c......      je(nel*ndofpe)=global dof column number for non-zero
c......      terms of each row (of element-dof connectivity info.)
c......     iet(ndof+1)=locations (pointers) of the first non-zero
c......     of each row (of dof-element connectivity info.)
c......      jet(nel*ndofpe)=locations (pointers) of the first non-zero
c......      of each row (of dof-element connectivity info.)
c......     ia(ndof)= ndof in the positions corespond to Dirichlet b.c.
c......            0  elsewhere
c......Output:ia(ndof+1)=starting locations of the first non-zero
c......     off-diagonal terms for each row of structural stiffness
c......     matrix
c......      ja(ncoeff)=column numbers (unordered) corespond to
c......      each nonzero, off-diagonal term of each row of structural
c......      stiffness matrix
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c "ia1" contains the supportdof from subroutine supportdof
c "ia2" is the new "ia" that gets set to zero at the start of this
c  subroutine and is created new at the end. GAH 4-23-04
c     print *,'*** ***  START SYMBASS  *** ***'
c                  PAUSE
c     print *,'*** ***  START SYMBASS  *** ***'
c                  PAUSE
c     print *,'*** ***  START SYMBASS  *** ***'
c                  PAUSE
```

```
c      print *,'*** *** START SYMBASS *** ***'
c                  PAUSE
c      print *,'ia(-) array at start symbass = ',(ia(i),i=1,ndof+1)
c                  PAUSE
c                   stop

                   n=ndof
     jp=1                    !001
     nm1=n-1                  !002
c                  print *,'nm1=',nm1
c                  print *,'n=',n
c                  print *,'ndof=',ndof
c                   print *,'ie(i) inside symbass=',(ie(i),i=1,nm1)
c                   print *,'je(i) inside symbass=',(je(i),i=1,nm1)
c                  stop
c                   print *,'ja(i) inside symbass=',(ja(i),i=1,nm1)
c                   print *,'jet(i) inside symbass=',(jet(i),i=1,nm1)
c                   print *,'iet(i) inside symbass=',(iet(i),i=1,nm1)
c                   print *,'ia1(i) start symbass=',(ia1(i),i=1,ndof+1)
c                   print *,'ia2(i) start symbass=',(ia2(i),i=1,ndof+1)
c                  pause
c initialize to zero
c                  do 9 k=1,ndof+1
c                  ia1(k)=0.
c                  ia2(k)=0.

c9                 continue
c                  stop
     do 30 i=1,nm1                !003 last row (= eq) will be skipped
     jpi=jp                     !004 delayed counter for ia(-) array
c     if ( ia2(i) .eq. n ) go to 30     !005 skip row which corespond to Dirichlet b.c.
     if ( ia(i) .eq. n ) go to 30     !005 skip row which corespond to Dirichlet b.c.

     ieta=iet(i)                !006 begin index (to find how many elements)
     ietb=iet(i+1)-1              !007 end index (to find how many elements)
      do 20 ip=ieta,ietb           !008 loop covering ALL elements attached to row i
      j=jet(ip)                 !009 actual "element number" attached to row i
      iea=ie(j)                 !010 begin index (to find how many nodes attached to
element j)
c                   print *,'ie(j)inside symbass=',ie(j)
c     print *,'ie(j+1)=',ie(j+1)
      ieb=ie(j+1)-1              !011 end index (to find how many nodes attached to
element j)
c                  print *,'iea & ieb=',iea,ieb
       do 10 kp=iea,ieb           !012 loop covering ALL nodes attached to element j
       k=je(kp)                 !013 actual "node, or column number" attached to element j
c                   print *,'k=',k
c                   print *,'i=',i
c                   print *,'ia2(k)=',ia2(k)
c                  pause
       if (k .le. i) go to 10       !014 skip, if it involves with LOWER triangular portion
       if ( ia(k) .ge. i ) go to 10  !015 skip, if same node already been accounted by earlier
elements
c        if ( ia2(k) .ge. i ) go to 10  !015 skip, if same node already been accounted by
earlier elements

       ja(jp)=k                  !016 record "column number" associated with non-zero off-
diag. term
       jp=jp+1                   !017 increase "counter" for column number array ja(-)
       ia(k)=i                   !018 record node (or column number) k already contributed
to row i
c        ia2(k)=i                  !018 record node (or column number) k already
contributed to row i

c                     print *,'ia2(k)=',ia2(k)
 10      continue                  !019
 20     continue                  !020
c     print *,'jpi = ',jpi
c                   pause
 30   ia(i)=jpi                   !021 record "starting location" of non-zero off-diag.
c 30   ia2(i)=jpi                   !021 record "starting location" of non-zero off-diag.

c                         !   terms associated with row i
c                   print *,'jp inside symbass=',jp
c                  pause
     ia(n)=jp                   !022 record "starting location" of non-zero term of LAST
ROW
     ia(n+1)=jp                  !023 record "starting location" of non-zero term of LAST
ROW + 1
c     ia2(n)=jp                   !022 record "starting location" of non-zero term of LAST
ROW
c     ia2(n+1)=jp                  !023 record "starting location" of non-zero term of LAST
ROW + 1

c
     ncoef1=ia(n+1)-1
c     ncoef1=ia2(n+1)-1
c                  print *,'ncoef1 inside symbass=',ncoef1
c                  pause
c                  print *,'ia(_) at end of symbass=',(ia(i),i=1,ndof+1)
c                  pause
c     write(6,*) 'ia(-) array = ',(ia(i),i=1,n+1)
c     write(6,*) 'ja(-) array = ',(ja(i),i=1,ncoef1)
c
     return
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c     Table 8.4  Complete FORTRAN Code for Sparse, Symmetrical
c              Numerical Assembly Process

     subroutine numass(ia,ja,idir,ae,be,lm,ndofpe,an,ad,b,ip,
    $elmass,dm,am,ianal,lumpmass,ndof,ncoef1)
c     subroutine numass(ia2,ja,idir,ae,be,lm,ndofpe,an,ad,b,ip,
c    $elmass,dm,am,ianal,lumpmass,ndof,ncoef1)
     implicit real*8(a-h,o-z)
c     dimension ia(*),idir(*),ae(*),be(*),lm(*),an(*)
     dimension ia(*),ja(*),idir(*),ae(*),be(*),lm(1),an(*)
c     dimension ia(*),idir(*),ae(*),be(*),lm(*),an(*)
c     dimension ia2(*),ja(*),idir(*),ae(*),be(*),lm(*),an(*)
     dimension ad(*),b(*),ip(*)
```

```
     dimension elmass(1),dm(1),am(1)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c......PLEASE direct your questions to Dr. Nguyen (nguyen@cee.odu.edu)
c......Purposes: symmetrical, sparse numerical assembly
c......         This code is stored under file name *symb*.f, in sub-directory
c......         ~
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c......Input: ia(ndof+1)=starting locations of the first non-zero
c......       off-diagonal terms for each row of structural stiffness
c......       matrix
c......        ja(ncoeff)=column numbers (unordered) corespond to
c......        each nonzero, off-diagonal term of each row of structural
c......        stiffness matrix
c......       idir(ndof)= 1 in the positions corespond to Dirichlet b.c.
c......                0 elsewhere
c......        ae(ndofpe**2),be(ndofpe)= element (stiffness) matrix,
c......        and element (load) vector
c......       lm(ndofpe)= global dof associated with a finite element
c......         ndofpe= number of dof per element
c......       b(ndof)= before using this routine, values of b(-) should
c......       be initialized to:
c......       Ci, values of prescribed Dirichlet bc at proper locations
c......       or  values of applied nodal loads
c
c......Output:  an(ncoeff1)= values of nonzero, off-diagonal terms of
c......         structural stiffness matrix
c......       ad(ndof)= values of diagonal terms of structural stiffness
c......       matrix
c......        b(ndof)= right-hand-side (load) vector of system of linear
c......        equations
c......Temporary Arrays:
c......       ip(ndof)= intialized to 0
c......              then IP(-) is used and reset to 0
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
c                  print *,'***** chk start numass *****'
c                  pause
c                  print *,'***** chk start numass *****'
c                  pause
c                  print *,'***** chk start numass *****'
c                  pause
c                  print *,'***** chk start numass *****'
c                  pause

c                  print *,'lm(i) start numass=',(lm(i),i=1,ndofpe)
c                  pause
c                  print *,'ndofpe inside numass =',ndofpe
c                  print *,'ndof inside numass=',ndof

c                  print *,'ad diag stiff start numass=',(ad(i),i=1,ndof)
c                  print *,'ae off diag stiff start numass=',(ae(i),i=1,ncoef1)
c     print *, 'an(i) array start numass = ',(an(i),i=1,50)
c     pause
c                  print *,'ia(-) start of numass = ',(ia(i),i=1,15+1)
c                  print *,'ja(-) start in numass = ',(ja(i),i=1,15+1)
c                  pause
c                  stop
c                  print *,'ae start of numass =',(ae(i),i=1,16)
C                  stop
c                  print *,'an start on numass =',(an(i),i=1,20)
c                  print *,'ad start on numass =',(ad(i),i=1,20)
c                  print *,'ip start in numass =',(ip(i),i=1,20)
c                  pause

c                  print *,'ia2(-) start of numass = ',(ia2(i),i=1,9+1)
c                  stop
c     print *,'ndofpe=',ndofpe
c                  print *,'idir=',(idir(ic),ic=1,ndof)
c                  pause

c     print *, 'ia2(-) array = ',(ia2(i),i=1,ndof+1)
c                  print *,'ncoef1=',ncoef1
c     print *, 'ja(-) array inside numass = ',(ja(ic),ic=1,ncoef1)
c                  print *,'ianal inside numass=',ianal
c                  print *,'idir=',(idir(i),i=1,ndof)
c                  print *,'b at start numass=',(b(i),i=1,ndof)
c                  pause
c                  print *,'be in numass=',(be(i),i=1,ndofpe)
c                  print *,'ia2=',(ia2(i),i=1,ndof)
c                  print *,'an=',(an(i),i=1,30)
c                  stop
c     write(*,*) 'ia2(-) start of numass = ',(ia2(i),i=1,ndof+1)
c     write(*,*) 'inside numass: b(-) = ',(b(i),i=1,10)
c                  stop
c     write(6,*) 'ia2(-) start of numass = ',(ia2(i),i=1,ndof+1)
c                  print *,'ia2(i) start numass=',(ia2(i),i=1,ndof+1)
c                  pause
c                  stop
c     write(6,*) 'inside numass: b(-) = ',(b(i),i=1,10)
     do 40 L=1,ndofpe               !001 local "row" dof
     i=lm(L)                     !002 global"row" dof
c      print *,'lm(L)=',lm(L),'  L=',L
c                   print *,'i=',i
c                   print *,'idir(i) in numass=',idir(i)
c                   pause
     if ( idir(i) .ne. 0 ) go to 401    !003 skip, if DIRICHLET b.c.
c      print *,'*** idir=0 ***'
                    k=L-ndofpe                  !004 to find location of element k-diag
c                   print *,'L=',L
c                   print *,'ndofpe=',ndofpe
c                   print *,'k=L-ndofpe',k
c                  print *,'ae(k+L*ndofpe)=',ae(k+L*ndofpe)
c                  pause
     ad(i)=ad(i)+ae(k+L*ndofpe)        !005 assemble K-diag

c                  print *,'ad in numass=',ad(i)
c                  pause
      if (ianal .eq. 1) then
      dm(i)=dm(i)+elmass(k+L*ndofpe)    !    assemble M-diag
      endif
c      print *,'assemble element rhs Load Vector in numass'
     b(i)=b(i)+be(L)                !006 assemble element rhs load vector
```

```
c     print *,'be(L) = ',be(L)
c                   print *,'b(i) = ',(b(ic),ic=1,ndof)
c                   stop
c                   pause

      kk=0                          !007 flag, to skip contribution of entire
c                                   !   global row i if all global col #
c                                   !   j < i, or if entire row i belongs
c                                   !   to LOWER triangle
      do 20 LL=1,ndofpe             !008 local "column" dof
c                   print *,'i two=',i
c                   print *,'ndofpe=',ndofpe
c                     print *,'LL=',LL
c                   pause
      k=k+ndofpe                    !009 find location of element stiffness k
c                   print *,'ndofpe=',ndofpe
c                   print *,'k=',k

c                   stop
      if (LL .eq. L) go to 20       !010 skip, diag. term already taken care
      j=lm(LL)                      !011 global column dof
c                   print *,'lm(LL)=',lm(LL)
c                   print *,'i three=',i
c                    print *,'j=',j
c                   pause
c                   print *,'idir(j)=',idir(j)
c                   pause
      if ( idir(j) .ne. 0 ) go to 10   !012 skip, if DIRICHLET b.c.
      if (j .lt. i) go to 20        !013 skip, if LOWER portion
      ip(j)=k                       !014 record global column # j (associated with
c                                   !   global row # i) corespond to k-th term
c                    print *,'ip(j)=',ip(j)
c                    print *,'j=',j
c                    print *,'i=',i
c                    pause
c                    write(6,*)'ip(j)=',ip(j)
c                    write(6,*)'j=',j
c                    write(6,*)'i=',i
c                   pause

c                   stop
c                                   !   of element stiffness k
      kk=1                          !015 FLAG, indicate row L of [k] do have
c                                   !   contribution to global row I of [K]
c                   print *,'kk in numass=',kk
c                   pause
      go to 20                      !016
 10   b(i)=b(i)-b(j)*ae(k)          !017 modify rhs load vector due to DIRICHLET b.c.

c                    print *,'ae(k)=',ae(k)
c                    print *,'i four=',i
c                    print *,'LL=',LL
c                   pause
c                    print *,'b(j)=',(b(j),j=1,ndof)

c                   stop
 20   continue
c                   print *,'Finished Do 20 in numass'
c                    print *,'b(i)=',(b(ic),ic=1,ndof)
c                    write(6,*)'kk=',kk
c                   print *,'kk=',kk
c                    stop
c                    print *,'L=',L
c                    print *,'i five=',i
c                   pause
      if (kk .eq. 0) go to 40       !019 skip indicator (see line 007)
      iaa=ia(i)                     !020 start index
      iab=ia(i+1)-1                 !021 end index
c                    write(6,*)'iaa=',iaa
c                    write(6,*)'iab=',iab
c                   print *,'i =',i
c                    print *,'ia(i)=',ia(i)
c                    print *,'ia2(i+1)=',ia2(i+1)
c                    print *,'iaa=',iaa
c                    print *,'iab=',iab
c                   pause
c                   stop
      do 30 j=iaa,iab               !022 loop covering all col numbers associated
c                                   !   with global row i
c                    print *,'j=',j
c                    print *,'ja(j) = ',ja(j)
c                   stop
c                    pause
      k=ip( ja(j) )                 !023 ip ( col # ) already defined on line 014
c                    print *,'k=',k

c                    write(6,*)'ja(j)=',ja(j)
c                    print *,'ja(j)=',ja(j)

c                    write(6,*)'k=',k
c                    print *,'ja(j)=',ja(j)

c                   pause
c                   print *,'j=',j
c                   print *,'an(j)=',an(j)
c                   print *,'iaa=',iaa
c                   print *,'iab=',iab
c                   stop
c                   pause
c                                   !            or initialized to ZERO initially
      if (k .eq. 0) go to 30        !024 skip, because eventhough row L of [k] do
c                                   !   have contribution to global row I of [K],
c                                   !   some terms of row L (of [k]) which associated
c                                   !   with DIRICHLET b.c. columns should be SKIPPED
c                   print *,'######### an & ae ##########'
c                   pause
c                   print *,'ae(k) in numass before=',ae(k)
c                   print *,'an(j) in numass before=',an(j)
c                   pause
      an(j)=an(j)+ae(k)             !025 assemble [K] from [k]

c                     write(6,*)'an(j)=',an(j)
c                     write(6,*)'ae(k)=',ae(k)
c     print *,'ae(k) in numass =',ae(k)
c                   print *,'######### an ##########'
c                   pause
c                   print *,'j in numass=',j
c                    print *,'an in numass=',an(j)
c                   pause
c                    stop

      if (ianal .eq. 1  .and.  lumpmass .eq. 0) then
      am(j)=am(j)+elmass(k)         !   assemble off-diag. [M] from [m]
      endif
      ip( ja(j) )=0                 !026 reset to ZERO for col # j before considering
c                                   !   the next row L
 30     continue                    !027
      go to 40
 401  continue
      ad(i)=1.0                     !   reset the diagonal of K(i,i)=1.0, due to b.c.
      if (ianal .eq. 1) dm(i)=1.0   !   reset the diagonal of M(i,i)=1.0, due to b.c.
c     write(6,*) 'inside numass: dirichlet dof#, ad(-) = ',i, ad(i)
c                    print *,'inside numass: dirichlet dof#, ad(-) = ',i, ad(i)
c                   stop
 40   continue                      !028
c     print *,'stop after 40'
c      print *, 'ad(i) diag stiff= ',(ad(ic),ic=1,ndof)
c      print *, 'an(i) off diag stiff = ',(an(ic),ic=1,50)
c                    pause
c                    stop
c......print debugging results
c     ndof=10
c     ncoefl1=ia(ndof+1)-1
c     write(6,*) 'at the end of routine numass'
c     write(6,*) 'ia2(-) array = ',(ia2(i),i=1,ndof+1)
c     write(6,*) 'ja(-) array = ',(ja(i),i=1,ncoef1)
c     write(6,*) 'ad(-) array = ',(ad(i),i=1,ndof)
c     write(6,*) 'an(i) array = ',(an(i),i=1,3)
c     write(6,*) ' b(-) array = ',( b(i),i=1,ndof)
c                   write(6,*) '*** END of numass ***'
c     print *, 'ia(-) array = ',(ia(i),i=1,ndof+1)
c     print *, 'ja(-) array = ',(ja(i),i=1,ncoef1)
c     print *, 'ad(-) array = ',(ad(i),i=1,ndof)
c     print *, 'an(i) array = ',(an(ic),ic=1,50)
c     print *, ' b(-) array end numass= ',( b(ic),ic=1,ndof)
c                   print *,'ad diag stiff end numass=',(ad(ic),ic=1,ndof)
c                   print *,'ae off diag stiff end numass=',(ae(i),i=1,ncoef1)

c                    print *,'at the end of routine numass'
c                   pause
c                    print *,'at the end of routine numass'
c                   pause
c                    print *,'at the end of routine numass'
c                   pause

c                   stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      subroutine materprop(nummat,prop,youngm)
      implicit real*8(a-h,o-z)
      dimension prop(1)
c
c......Purposes: Input material properties, assuming max. 10 materials,
c            and each material type has 11 properties
c......Notes:   This routine can be used for any finite element type !
c
      nprop=11
       do 1 i=1,nummat
       istart=(i-1)*nprop
       read(5,*) e1,e2,e3,xnu12,xnu23,xnu31,g12,g23,g31,rho,thermoc
       write(6,*) 'e1,e2,e3,xnu12,xnu23,xnu31,g12,g23,g31,rho,thermoc'
       write(6,*)  e1,e2,e3,xnu12,xnu23,xnu31,g12,g23,g31,rho,thermoc
       prop(istart+1)=e1
       prop(istart+2)=e2
       prop(istart+3)=e3
       prop(istart+4)=xnu12
       prop(istart+5)=xnu23
       prop(istart+6)=xnu31
       prop(istart+7)=g12
       prop(istart+8)=g23
       prop(istart+9)=g31
       prop(istart+10)=rho
       prop(istart+11)=thermoc
 1     continue

                    youngm=e1

c                   print *,'e1=',e1
c                   print *,'e2=',e2
c                  stop

      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      subroutine sectprop(nsect,prop,area)
      implicit real*8(a-h,o-z)
      dimension prop(1)
c
c......Purposes: Input cross-sectional properties, assuming max. 10 sections,
c            and each cross-sectional type has 12 properties
c......Notes:   This routine can be used for any finite element type !
c
      nprop=12
       do 1 i=1,nsect
       istart=(i-1)*nprop
       read(5,*) area,xi,yi,zi,t1,t2,t3,t4,
     $theta1,theta2,theta3,theta4
       write(6,*) 'area,xi,yi,zi,t1,t2,t3,t4'
       write(6,*) 'theta1,theta2,theta3,theta4'
       write(6,*)  area,xi,yi,zi,t1,t2,t3,t4
       write(6,*)  theta1,theta2,theta3,theta4
```

```
      prop(istart+ 1)=area
      prop(istart+ 2)=xi
      prop(istart+ 3)=yi
      prop(istart+ 4)=zi
      prop(istart+ 5)=t1
      prop(istart+ 6)=t2
      prop(istart+ 7)=t3
      prop(istart+ 8)=t4
      prop(istart+ 9)=theta1
      prop(istart+10)=theta2
      prop(istart+11)=theta3
      prop(istart+12)=theta4
 1    continue

                    area = prop(istart+1)

c                   print *,'area inside sectprop=',area
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine nodecoor(numnodes,xxx,yyy,zzz)
      implicit real*8(a-h,o-z)
c
c......Purpose: Input joint coordinates
c
      dimension xxx(1),yyy(1),zzz(1)
c
c                   print *,'numnodes inside nodecoor=',numnodes

      write(6,*) 'joint,xxx(joint),yyy(joint),zzz(joint)'
      do 21 i=1,numnodes
      read(5,*) joint,xxx(joint),yyy(joint),zzz(joint)
c                   write(*,4)  joint,xxx(joint),yyy(joint),zzz(joint)
c                   pause
      write(6,4)  joint,xxx(joint),yyy(joint),zzz(joint)
 4    format(2x,i5,f10.2,f10.2,f10.2)
 21   continue
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine elconect(ie,je,lm,neltype,nel,ndofpe,ndofpn)
      implicit real*8(a-h,o-z)
c
c......Purpose: Input element connectivity information.
c......Note: This routine can be used for any finite element type ??
c
      dimension ie(1),je(1),lm(1),nel(1),ndofpe(1)
c
       ieadd=0
       locadd=0
      ie(1)=1                  ! always
c                   print *,'neltype inside elconect=',neltype

c                   pause
c                   stop
      do 21 k=1,neltype
      numel=nel(k)
c                   print *,'numel inside elconect=',numel
c                   pause
      if (numel .eq. 0) go to 21
      go to (31,32,33,34,34,34),k
 31   continue                 ! 2-d truss
c                   print *,'numel=',numel
c                   pause
      do 22 i=1,numel
      ie(i+1)=ie(i)+ndofpe(k)
c                   print *,'ie(i+1) inside elconect =',ie(i+1)
c                   pause
c
      read(5,*) ithel,nodei,nodej
c                   print *,'Read inside elconect'
c     write(6,*) 'ithel,nodei,nodej = ',ithel,nodei,nodej
c     print *,'ithel,nodei,nodej = ',ithel,nodei,nodej
c                   pause
c Use for 2D truss
c     lm(1)=nodei*ndofpn-1
c     lm(2)=lm(1)+1
c     lm(3)=nodej*ndofpn-1
c     lm(4)=lm(3)+1
c                   print *,'nodei=',nodei
c                   print *,'ndofpn=',ndofpn
c Use below for 3D Truss
      lm(1)=nodei*ndofpn-2
      lm(2)=lm(1)+1
                    lm(3)=lm(1)+2
      lm(4)=nodej*ndofpn-2
      lm(5)=lm(4)+1
      lm(6)=lm(4)+2
      locbegin=(ithel-1)*ndofpe(k)
c                   print *,'ithel=',ithel
c                   print *,'nodei=',nodei
c                   print *,'nodej=',nodej
c                   print *,'ndofpn=',ndofpn
c                   print *,'lm(1)=',lm(1)
c                   print *,'lm(2)=',lm(2)
c                   print *,'lm(3)=',lm(3)
c                   print *,'lm(4)=',lm(4)
c                   print *,'lm(5)=',lm(5)
c                   print *,'lm(6)=',lm(6)
c                   print *,'locbegin=',locbegin
c                   pause
       do 23 j=1,ndofpe(k)
       locate=locbegin+j
       je(locate)=lm(j)
c                   print *,'locate=',locate
c                   print *,'je(locate)=',je(locate)
c                   pause
c                     print *,'lm(j)=',lm(j)
c                     print *,'je(locate)=',je(locate)
c       pause
 23     continue
 22   continue
      go to 21
 32   continue                 ! 3-d beam
      ieadd=ieadd+nel(k-1)       ! to include previous 2-d truss elements
      locadd=locadd+nel(k-1)*4
      go to 21
 33   continue                 ! 4-node (24 dofpe) rect. plate bending+membrane
      ieadd=ieadd+nel(k-1)       ! to include previous 3-d beam elements
      locadd=locadd+nel(k-1)*12
c                   print *,'numel=',numel
c                   pause
      do 42 i=1,numel
      ie(i+1+ieadd)=ie(i+ieadd)+ndofpe(k)
c
      read(5,*) ithel,nodei,nodej,nodek,nodel
      write(6,*) 'ithel,nodei,nodej,nodek,nodel'
      write(6,*)  ithel,nodei,nodej,nodek,nodel
c
      lm(1)=nodei*ndofpn-5
      lm(2)=lm(1)+1
      lm(3)=lm(2)+1
      lm(4)=lm(3)+1
      lm(5)=lm(4)+1
      lm(6)=lm(5)+1
c
      lm( 7)=nodej*ndofpn-5
      lm( 8)=lm( 7)+1
      lm( 9)=lm( 8)+1
      lm(10)=lm( 9)+1
      lm(11)=lm(10)+1
      lm(12)=lm(11)+1
c
      lm(13)=nodek*ndofpn-5
      lm(14)=lm(13)+1
      lm(15)=lm(14)+1
      lm(16)=lm(15)+1
      lm(17)=lm(16)+1
      lm(18)=lm(17)+1
c
      lm(19)=nodel*ndofpn-5
      lm(20)=lm(19)+1
      lm(21)=lm(20)+1
      lm(22)=lm(21)+1
      lm(23)=lm(22)+1
      lm(24)=lm(23)+1
c
      locbegin=(ithel-1)*ndofpe(k)
       do 43 j=1,ndofpe(k)
       locate=locbegin+j
       je(locate+locadd)=lm(j)
 43     continue
 42   continue
      go to 21
 34   continue                 ! other f.e. type
      go to 21

 21   continue
c
      ieall=nel(1)+nel(2)+nel(3)+nel(4)+nel(5)+nel(6)
cc      jeall=nel(1)*4 + nel(2)*12 + nel(3)*24
      jeall=nel(1)*6 + nel(2)*12 + nel(3)*24
c                   print *,'jeall=',jeall
c                   pause
c
c     write(6,*) 'ieall = sum all el types = ',ieall
c     write(6,*) 'ie= ', (ie(i),i=1,ieall+1)
c     write(6,*) 'je = ',  (je(i),i=1,jeall)
c     print *, 'ieall = sum all el types = ',ieall
c                   print *, 'jeall = ',jeall
c                   print *, 'ie= ', (ie(i),i=1,ieall+1)
c     print *, 'je = ',  (je(i),i=1,jeall)
c                   pause
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      subroutine loads(ndof,b,loadof,exfor)
      implicit real*8(a-h,o-z)
c
c......Purpose: Input applied loads at the joints
c
      dimension b(1),exfor(*)
c
c      write(6,*) 'inside loads: ndof = ',ndof
c                   print *,'inside loads'
      do 1 i=1,ndof
      b(i)=0.0    ! load vector
 1    continue

       do 18 i=1,loadof
       read(5,*) loadedof, value
       write(6,*) 'loadedof, value'
       write(6,*)  loadedof, value

c                   print *,'loadof=',loadof
c                   print *,'value=',value
c                   pause

       b(loadedof)=value
                    exfor(loadedof)=value
 18    continue
c                   print *,'exfor inside loads=',(exfor(i),i=1,ndof)
c                   stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      subroutine supportdof(iboundc,nboundc,b,ndof,ia)
      implicit real*8(a-h,o-z)
c
```

```
c......Purpose: input support dof information
c
      dimension iboundc(1),       b(1),ia(1)

      do 18 i=1,ndof
      iboundc(i)=0
 18   continue
c......input the boundary condition flag
        do 19 i=1,nboundc
        read(5,*) ibcdof, settlem
        write(6,*) 'ibcdof, settlem = ',ibcdof,settlem
        iboundc(ibcdof)=1
        b(ibcdof)=settlem
c                    print *,'b(i) inside supportdof=',b(i)
 19     continue
c                    pause
c     write(6,*) ' ibcdof = ', ibcdof
      write(6,*) ' load and settlement b(-) = ', (b(i),i=1,ndof)
c......copy boundary flag for later usage in Duc's sparse symbolic assembly
      do 3 i=1,ndof
      if ( iboundc(i) .eq. 0 ) ia(i)=0
      if ( iboundc(i) .eq. 1 ) ia(i)=ndof
 3    continue
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine assembly(ip,ad,an,lm,ie,je,propmat,
c    $         propsect,elk,be,b,iboundc,
     $         propsect,be,b,iboundc,
     $         ae,ia,ja,xx,yy,zz,ncoef1,maxdofpc,
c    $         ae,ia,ja,ncoef1,maxdofpe,
c    $         ae,ia2,ja,xx,yy,zz,ncoef1,maxdofpc,
     $ ndof,neltype,nel,ndofpc,ianal,lumpmass,dm,am,elm,
c    $ tempo1,disp,iarc,it,area,youngm,esm,qk,itnrmlon)
     $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon,istep,icheck,
     $ zee,dLength,icx)

      implicit real*8(a-h,o-z)
c
c     Purpose: perform sparse numerical assembly
c
c
      dimension ip(1),ad(1),an(1),lm(1),ie(1),je(1),propmat(1)
     $,      propsect(1),be(1),b(1),iboundc(1)
c    $,      propsect(1),elk(maxdofpe,maxdofpe),be(1),b(1),iboundc(1)
     $,   ae(1),ia(1),ja(1),xx(500),yy(500),zz(500),nel(1),ndofpe(1)
c    $,      ae(1),ia2(1),ja(1),xx(1),yy(1),zz(1),nel(1),ndofpe(1)
      dimension dm(1),am(1),elm(maxdofpe,1),tempo1(1)

     common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
     common/disp/ux(2),uy(2),uz(2),P(6),B1(6)

                    common/jam/ja2(100)
c     common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)

c                    dimension ux(2),uy(2),finx(2),finy(2)
                     dimension disp(*),qk(*)

                     dimension lm2(6)

                     dimension elk(6,6)

c                   print *,'je start assembly=',(je(ic),ic=1,10)
c                   pause
c                   print *,'B(_) start assembly=',(b(ic),ic=1,ndof)
c                   pause

c     double precision, allocatable:: b(:,:)
c                   allocate elk(maxdofpe,maxdofpe)

c                    dimension ja(1)

c                   print *,'Getstiff start assembly=',getstiff
c                   print *,'Getforce start assembly=',getforce
c                   print *,'Getsolve start assembly=',getsolve
c                   pause

                    If(getsolve .eq. 1)Then
                    Goto 670
                    Endif

                    If(getforce .eq. 1)Then !Skip ad & ae =0,just getforce
                    Goto 699
                    Endif

c                   print *,'S T A R T   O F   A S S E M B L Y'
c                   print
*,'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
c                   print *,'Setting ad, an & ae to zero'
      Do 701 ic=1,ndof
                     ad(ic)=0.0 !Stiffness diagonal

c                    an(ic)=0.0 !Stiffness off diagonal

701    continue

                     Do 702 ic=1,10000
                     an(ic)=0.d0
702    continue


c                   print *,'b(_) at start2 of assembly=',(b(ic),ic=1,ndof)
c                   pause
c                     Do 702 ic=1,100
c                     ae(ic)=0.0
c702     continue
c                   print
*,'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
c                   pause
699                 Continue
```

```
c                   print *,'idir at start of assembly=',(idir(ic),ic=1,ndof)
c                   pause

c                   print *,'qk(_) at start of assembly=',(qk(ic),ic=1,ndof)
c                   print *,'b(_) at start of assembly=',(b(ic),ic=1,ndof)
c                   pause
c                   print *,'ia start of assembly=',(ia(i),i=1,15)
c                   print *,'ja2(i)=',(ja2(ic),ic=1,10)
c                   print *,'ja(1)=',ja(1)
c                   print *,'ja start of assembly=',(ja(i),i=1,1)
c                   print *,'ndofpe = ',ndofpe
c                   stop

c                   If(getforce .eq. 1)then
c                   Do 777 i = 1,10
c                   ja(i)=ja2(i)
c                   Print *,'ja2(i) inside start assembly=',ja2(i)
c777                Continue
c                   pause
c                   Endif

c                   if(it .eq. 2)then
c                   print *,'it =2'
c                   print *,'ianal=',ianal
c                   stop
c                   endif
c                   print *,'getforce=',getforce
c                   stop
c                   print *,'chk inside assembly at start'
c                   print *,'getstiff=',getstiff
c                   print *,'getforce=',getforce
c                   pause
c                   print *,'xx inside assembly = ',(xx(i),i=1,42)
c                   print *,'yy inside assembly = ',(yy(i),i=1,42)
c                   print *,'zz inside assembly = ',(zz(i),i=1,42)
c                   print *,'lm inside assembly =',(lm(i),i=1,6)
c                   print *,'____________________________________________'
c                   pause
c                   stop
c
c                   pause

c                    If(getforce .eq. 1)then
c                    print *,'getforce=1 at start of assembly'
c                    stop
c                    endif

c                   write(6,*)'ia2 start of assembly=',(ia2(i),i=1,ndof+1)
c                    print *,'chk inside assembly'
c                   print *,'iarc inside assembly=',iarc
c                   print *,'ianal inside assembly=',ianal
c                   print *,'neltype=',neltype
c                                       print *,'ncoef1=',ncoef1
c                   print *,'ndof=',ndof
cc                  ncoef2=ncoef1
c                   print *,'ip=',(ip(i),i=1,ndof)
c                   stop
c                   print *,'b=',(b(i),i=1,ndof)
c                   pause
c                    print *,'ia2 start assembly=',(ia2(i),i=1,ndof+1)
c                   stop

c......initialize before numerical sparse assembling
      do 96 j=1,ndof
96     ip(j)=0
cccc blanked out below don't need dynamics
cccc       if (ianal .eq. 1) dm(j)=0.
cccc 96    ad(j)=0.
c 96    continue
c
cccc something not write with ncoef1???
cc                  ncoef1=ncoef2

c                    write(6,*)'ncoef1 start assembly=',ncoef1
c     print *,'ncoef1=',ncoef1
c                   stop
cccc      do 95 j=1,ncoef1
cccc       if (ianal .eq. 1  .and. lumpmass .eq. 0) then
cccc       am(j)=0.
cccc       endif
cccc 95    an(j)=0.
c                   print *,'neltype=',neltype
c                   stop

c                   if(it .eq. 2)then
c                   print *,'it =2'
c                   print *,'neltype=',neltype
c                   stop
c                   endif

c                    write(6,*),'neltype in assembly=',neltype
c                   stop

670                 Continue !skipped above when getsolve=1
c**** START OF DO 98 ******************************
      DO 98 k=1,neltype

                    If(getsolve .eq. 1)Then
                    Goto 671
                    Endif

      numel=nel(k)
c                                 print *,'numel=',numel
c                   pause
c                   if(it .eq. 2)then
c                   print *,'it =2'
c                   print *,'ianal=',ianal
c                   print *,'numel=',numel
c                   stop
c                   endif

c                       stop
```

```
      if (numel .eq. 0) go to 98
c.....loop for all elements
c                     print *,'start loop for all elements'
c                           print *,'ncoef1=',ncoef1
c                       print *,'numel =',numel
c                     pause
c                     stop

      DO 4 iel=1,numel
c                     print *,'iel = ',iel
c                     print *,'********************'

c                     If(getforce .eq. 1)then
c                     goto 400
c                     endif

c                     pause
c                     print *,'numel=',numel
c                     pause
c                     stop
c     write(6,*) 'inside assembly: truss el #,nel(1) = ',iel,nel(1)
c.....get the global dof associated with each element
c                     print *,'ndofpe(k)=',ndofpe(k)
      iend=iel*ndofpe(k)
      istart=iend-ndofpe(k)
c                     print *,'iend=',iend
c                     print *,'istart=',istart
c                     print *,'ndofpe(k)=',ndofpe(k)
c                     pause

      do 76 j=1,ndofpe(k)
c                     print *,'j=',j
c                     print *,'je(istart+j)=',je(istart+j)
c                      write(6,*)'lm(j)=',lm(j)
      lm(j)=je(istart+j)
 76                   continue
c 76                  print *,'je(istart+j)=',je(istart+j)
c                     print *,'lm 2 inside assembly =',(lm(i),i=1,6)
c     pause

c                     print *,'iarc=',iarc
c                     pause
c                     stop

                      if(iarc .eq. 1)then
                      ix1dof=lm(1)
                      iy1dof=lm(2)
                      iz1dof=lm(3)
                      ix2dof=lm(4)
                      iy2dof=lm(5)
                      iz2dof=lm(6)
c                     print *,'lm(6)=',lm(6)
c     ux(1)=ux(1)+disp(ix1dof)
c                     uy(1)=uy(1)+disp(iy1dof)
c                     uz(1)=uz(1)+disp(iz1dof)
c                     ux(2)=ux(2)+disp(ix2dof)
c                     uy(2)=uy(2)+disp(iy2dof)
c                     uz(2)=uz(2)+disp(iz2dof)
c     ux(1)=ux(1)+qk(ix1dof)
c                     uy(1)=uy(1)+qk(iy1dof)
c                     uz(1)=uz(1)+qk(iz1dof)
c                     ux(2)=ux(2)+qk(ix2dof)
c                     uy(2)=uy(2)+qk(iy2dof)
c                     uz(2)=uz(2)+qk(iz2dof)
c                     print *,'qk(ix1dof)=',qk(ix1dof)
c                     stop
     ux(1)=qk(ix1dof)
                      uy(1)=qk(iy1dof)
                      uz(1)=qk(iz1dof)
                      ux(2)=qk(ix2dof)
                      uy(2)=qk(iy2dof)
                      uz(2)=qk(iz2dof)
c                     print *,'ix1dof=',ix1dof
c     print *,'iy1dof=',iy1dof
c     print *,'iz1dof=',iz1dof
c     print *,'ix2dof=',ix2dof
c     print *,'iy2dof=',iy2dof
c     print *,'iz2dof=',iz2dof
c     print *,'ux(1)=',ux(1)
c     print *,'uy(1)=',uy(1)
c     print *,'uz(1)=',uz(1)
c     print *,'ux(2)=',ux(2)
c     print *,'uy(2)=',uy(2)
c     print *,'uz(2)=',uz(2)
c                     pause
c                     stop

                      If(getforce .eq. 1)then
c                      print *,'ix1dof=',ix1dof
c     print *,'iy1dof=',iy1dof
c     print *,'iz1dof=',iz1dof
c     print *,'ix2dof=',ix2dof
c     print *,'iy2dof=',iy2dof
c     print *,'iz1dof=',iz1dof
c     print *,'ux(1)=',ux(1)
c     print *,'uy(1)=',uy(1)
c     print *,'uz(1)=',uz(1)
c     print *,'ux(2)=',ux(2)
c     print *,'uy(2)=',uy(2)
c     print *,'uz(2)=',uz(2)
c                      write(6,*),'ix1dof=',ix1dof
c     write(6,*),'iy1dof=',iy1dof
c     write(6,*),'iz1dof=',iz1dof
c     write(6,*),'ix2dof=',ix2dof
c     write(6,*),'iy2dof=',iy2dof
c     write(6,*),'iz2dof=',iz2dof
c     write(6,*),'ux(1)=',ux(1)
c     write(6,*),'uy(1)=',uy(1)
c     write(6,*),'uz(1)=',uz(1)
c     write(6,*),'ux(2)=',ux(2)
c     write(6,*),'uy(2)=',uy(2)
c     write(6,*),'uz(2)=',uz(2)
c                     stop
                      endif

                      endif

c                     print *,'chk 2 inside assembly'
c                     stop
c                     pause

c                     if(it .eq. 2)then
c                     print *,'it =2'
c                     print *,'ianal=',ianal
c                     stop
c                     endif

c
c                     print *,'chk before info_elstif inside assembly'
c                     pause
c                     print *,'lm 3 inside assembly =',(lm(i),i=1,6)

c                     print *,'xx inside assembly before info_= ',(xx(i),i=1,42)
c                     print *,'yy inside assembly before info_= ',(yy(i),i=1,42)
c                     print *,'zz inside assembly before info_= ',(zz(i),i=1,42)
c                     pause
c                     print *,'**** **** **** **** **** ****'
c                     print *,'before "call info_elstif"  '
c                     pause

c                     print *,'xlength before info_elstif=',xlength
c                     print *,'xli before info_elstif=',xli
c                     pause
c                     print *,'je before info_elstif=',(je(ic),ic=1,10)
c                     pause
c                     print *,'B(_) before info_elstif in assembly=',(b(ic),ic=1,ndof)
c                     pause
                      If(icx .eq. 10)then
                      goto 1900
c                     call info_ex(iel,xlength,xl,cx,cy,cz,xx,yy,zz,ic,je
c     $,propmat,propsect,youngm,area,rho,
c     $ deltaxi,deltayi,deltazi,xli,
c     $ xi,xj,yi,yj,zi,zj,AE,zee,dLength)
                       Else
      call info_elstif(iel,xlength,xl,cx,cy,cz,xx,yy,zz,ic,je
     $,propmat,propsect,youngm,area,rho,
     $ deltaxi,deltayi,deltazi,xli,
     $ xi,xj,yi,yj,zi,zj)
                      Endif

1900  CONTINUE !dummy

c                     print *,'xx inside assembly after info_= ',(xx(i),i=1,42)
c                     print *,'yy inside assembly after info_= ',(yy(i),i=1,42)
c                     print *,'zz inside assembly after info_= ',(zz(i),i=1,42)
c                     pause
c                     print *,'**** **** **** **** **** ****'
c                     print *,'after "call info_elstif"  '
c                     print *,'B(_) after info_elstif in assembly=',(b(ic),ic=1,ndof)
c                     pause

c                     print *,'xlength after into_elstif=',xlength
c                     print *,'xli after info_elstif=',xli
c                     pause

      ndofped=ndofpe(k)
c     write(6,*) 'inside assembly: ndofped = ',ndofped
c                     print *,'chk before clstif inside assembly'
c                     stop
c                     print *,'lm 3a inside assembly =',(lm(i),i=1,6)

c                     pause
c      call elstif(ndofped,elk,be,
c     $           area,youngm,xlength,cx,cy,cz,maxdofpc,
c     $           ityp,xl,xli,deltaxi,deltayi,deltazi,
c     $           finx,finy,finz,iarc)
c      call elstif(ndofpe,esm,be,

c                     print *,'xx inside assembly before elstif= ',(xx(i),i=1,42)
c                     print *,'yy inside assembly before elstif= ',(yy(i),i=1,42)
c                     print *,'zz inside assembly before elstif= ',(zz(i),i=1,42)
c                     print *,'++++++++++++++++++++++++++++++++++++++++++++'
c                     print *,'Before elstif inside assembly'
c                     pause
c                     print *,'xlength before elstif=',xlength
c                     print *,'xli before elstif=',xli
c                     pause

c                     print *,'iel before call elstif=',iel
c                     pause

      call elstif(ndofpe,elk,be,
     $           area,youngm,xlength,cx,cy,cz,maxdofpe,
     $           xl,xli,deltaxi,deltayi,deltazi,xx,yy,zz,
     $         iarc,xi,xj,yi,yj,zi,zj,ityp,tm,istep,iel,icheck,iex)

c                     print *,'xlength after elstif=',xlength
c                     print *,'xli after elstif=',xli
c                     pause

c                     print *,'xx inside assembly after elstif= ',(xx(i),i=1,4)
c                     print *,'yy inside assembly after elstif= ',(yy(i),i=1,4)
c                     print *,'zz inside assembly after elstif= ',(zz(i),i=1,4)
c                     print *,'++++++++++++++++++++++++++++++++++++++++++++'
c                     print *,'After elstif inside assembly'
c                     pause

c                     print *,'lm 3b inside assembly =',(lm(i),i=1,6)
c                     pause
c                      print *,'Check after elstif'

c                     print *,'idir after elstif in assembly=',(idir(ic),ic=1,ndof)
c                     pause
```

```
c                 DO 919 i=1,6
c                 DO 919 j=1,6
c919              print *,'elk after elstif = ',elk(i,j)
c                 pause
c     stop
c                 print *,'getforce=',getforce
c                 print *,'finx(1)=',finx(1)
c                 print *,'finy(1)=',finy(1)
c                 print *,'finz(1)=',finz(1)
c                 print *,'finx(2)=',finx(2)
c                 print *,'finy(2)=',finy(2)
c                 print *,'finz(2)=',finz(2)
c                 pause
c                 stop

                  If(getforce .eq. 1)then
                    b(ix1dof)=finx(1)+b(ix1dof)
                    b(iy1dof)=finy(1)+b(iy1dof)
                    b(iz1dof)=finz(1)+b(iz1dof)
                    b(ix2dof)=finx(2)+b(ix2dof)
                    b(iy2dof)=finy(2)+b(iy2dof)
                    b(iz2dof)=finz(2)+b(iz2dof)
c                 write(6,*)'b x&y=',(b(i),i=1,ndof)
c                 print *,'b x,y,z=',(b(i),i=1,ndof)
c                 pause
c                 stop
c                 goto 4
      endif

c                 print *,'lm 4 inside assembly =',(lm(i),i=1,6)
c                 pause
c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 print *,'ianal=',ianal
c                 stop
c                 endif
c                 print *,'ianal =',ianal

      if (ianal .eq. 1) then
      call elmass(rho,area,xlength,lumpmass,elm,ndofped,maxdofpe,
     $cx,cy)
      endif
c
c     write(6,*) '*** passed elstiff ***'
c                  print *,'*** passed elstiff ***'
c                 print *,'ndofpe(k)=',ndofpe(k)
c                 pause
c                 stop

c                 DO 920 i=1,6
c                 DO 920 j=1,6
c920              print *,'elk after elstif = ',elk(i,j)

c                 print *,'zz 1 inside assembly after elstif= ',(zz(ic),ic=1,3)
c                 pause


c677   Continue !skip to here from start if getsolve=1, just solving

c.....convert element [k] & [m] from 2-D array into 1-D array (column-wise)
      do 78  j=1,ndofpe(k)
c                 print *,'lm 5 inside assembly =',(lm(icc),icc=1,6)

c                 print *,'zz 1a inside assembly after elstif= ',(zz(ic),ic=1,3)
c                 pause
      do 77 jj=1,ndofpe(k)
c                 print *,'lm 6 inside assembly =',(lm(ic),ic=1,6)
      locate=jj+(j-1)*ndofpe(k)
c                 print *,'lm 6a inside assembly =',(lm(ic),ic=1,6)

c                 print *,'zz 1b inside assembly after elstif= ',(zz(ic),ic=1,3)
c                 pause
cc                print *,'ae(locate)=',ae(locate)

ccc Something wrong with lm ??????????????????????????
c                 do 812 im=1,6
c812              lm2(im)=lm(im)

c                 print *,'zz 1c inside assembly after elstif= ',(zz(id),id=1,3)
c                 pause
c                 print *,'jj, j, elk(jj,j)=',jj,'  ',j,'  ',elk(jj,j)
c                 pause
c                  print *,'locate, ae(locate) before=',locate,'  ',ae(locate)
c                 pause
      ae(locate)=elk(jj,j)
c                  print *,'locate, ae(locate) after=',locate,'  ',ae(locate)
c                 pause
c                 print *,'zz 1d inside assembly after ae(_)= ',(zz(ic),ic=1,3)

c                 Do 922 id=1,3
c                 zz(id)=0.0 !Force zz to zero, something wrong above!!!
c922   continue
c                 print *,'zz 1d forcing to zero= ',(zz(ic),ic=1,3)

c                  pause

c                 do 813 im=1,6
c813   lm(iun)=lm2(im)
                        !  for [k]
cc                print *,'lm 6b inside assembly =',(lm(ic),ic=1,6)
c                  write(6,*)'locate=',locate
c                  write(6,*)'ae(locate)=',ae(locate)
c       print *,'locate end assembly=',locate
c                    print *,'ae(locate) end assembly=',ae(locate)
c                 pause
c                    print *,'jj, j = ',jj,'  ',j
c                    print *,'elk(jj,j)=',elk(jj,j)
c                 pause
c                  write(6,*)          'jj, j, locate        = ',jj,'  ',j,' ',locate
c                  write(6,*)'elk(jj,j)=',elk(jj,j)
c                    pause
      if (ianal .eq. 1) then
      tempo1(locate)=elm(jj,j)    !  for [m]
c                                 write(6,*)'tempo1=',tempo1
      endif
cc                print *,'lm 7 inside assembly =',(lm(ic),ic=1,6)
c                 pause
c                 print *,'zz 2 inside assembly lm 7= ',(zz(icc),icc=1,3)
c                 pause
 77   continue
 78   continue

c                 print *,'ae(_)=',(ae(ic),ic=1,20)
c                 pause
c
c                 print *,'lm 8 inside assembly =',(lm(ic),ic=1,6)
c                   stop
c                 print *,'going to numass'
c                 print *,'ncoef1=',ncoef1
c                 print *,'ia before numass in assembly=',(ia(ic),ic=1,10+1)
c                 pause
c not correct...something wrong with ncoef1, again
c                 ncoef2=ncoef1
c                 stop
c.....sparse "numerical" assembly
c                  print *,'ia1 before numass in assembly=',(ia1(i),i=1,ndof+1)
c                  print *,'ia2 before numass in assembly=',(ia2(i),i=1,ndof+1)
c                 write(6,*)'ia2 before numass in assembly=',(ia2(i),i=1,ndof+1)
c                 stop
c                 print *,'ja(1) before numass in assembly=',ja(1)
c                 pause
c                 print *,'lm inside assembly before numass =',(lm(i),i=1,6)

c                 print *,'iboundc before numass in ably=',(iboundc(ic),ic=1,ndof)
c                 pause
c...iboundc=idir in subroutine numass
c...tempo1=elmass in subroutine numass

c                 print *,'xx inside assembly before numass= ',(xx(i),i=1,42)
c                 print *,'yy inside assembly before numass= ',(yy(i),i=1,42)
c                 print *,'zz inside assembly before numass= ',(zz(i),i=1,42)
c                 print *,'++++++++++++++++++++++++++++++++++++++++'
c                 print *,'++++++++++++++++++++++++++++++++++++++++'
c                 print *,'Before numass inside assembly'
c                 pause

c     print *, ' b(-) before numass in assembly= ',( b(i),i=1,ndof)
c                 pause
c                 print *,'getforce=',getforce
c                 pause

                  If(getforce .eq. 1)Then
c                 print *,'getforce = 1, skip numass'
                  goto 405
                  Endif

671   Continue !skip to here from start if getsolve=1, just solving

c     print *, '*** before numass ***'
c     print *,'ad(-) = ',(ad(i),i=1,ndof)
c     print *, 'an(-) = ',(an(i),i=1,ncoef1)
c     print *, ' b(-) before numass in assembly= ',( b(i),i=1,ndof)
c                 print *,'iboundc before numass=',(iboundc(ic),ic=1,ndof)
c                 pause

     call numass(ia,ja,iboundc,ac,bc,lm,ndofped,an,ad,b,ip,
    $tempo1,dm,am,ianal,lumpmass,ndof,ncoef1)
c     print *, ' b(-) after numass in assembly= ',( b(i),i=1,ndof)
c                 pause
405   continue !dummy line

c                 print *,'xx inside assembly after numass= ',(xx(i),i=1,42)
c                 print *,'yy inside assembly after numass= ',(yy(i),i=1,42)
c                 print *,'zz inside assembly after numass= ',(zz(i),i=1,42)
c                 print *,'++++++++++++++++++++++++++++++++++++++++'
c                 print *,'++++++++++++++++++++++++++++++++++++++++'
c                 print *,'After numass inside assembly'
c                 pause

c     call numass(ia2,ja,iboundc,ae,be,lm,ndofped,an,ad,b,ip,
c    $tempo1,dm,am,ianal,lumpmass,ndof,ncoef1)
c     write(6,*) '*** passed numass ***'
c     write(6,*) 'ad(-) = ',(ad(i),i=1,ndof)
c     write(6,*) 'an(-) = ',(an(i),i=1,ncoef1)
c     write(6,*) ' b(-) = ',( b(i),i=1,ndof)
c     print *, '*** passed numass ***'
cc     print *,'ad(-) = ',(ad(i),i=1,ndof)
cc     print *, 'an(-) = ',(an(i),i=1,ncoef1)
c     print *, ' b(-) after numass in assembly= ',( b(i),i=1,ndof)
c                 pause
c                  print *,'ncoef1 after numass=',ncoef1
c                 stop

c                  print *,'ia2 after numass in assembly=',(ia2(i),i=1,ndof+1)
c                 stop
c                  ncoef1=ncoef2
c                  print *,'ncoef1=',ncoef1
c     print *, 'ad diag stiff = ',(ad(i),i=1,ndof)
c     print *, 'an off diag stiff = ',(an(i),i=1,ncoef1)
c                 stop
c     write(6,*) '*** passed numass ***'
c                  write(6,*) 'iel =',iel
c     write(6,*) 'ad(-) = ',(ad(i),i=1,ndof)
c     write(6,*) 'an(-) = ',(an(i),i=1,ncoef1)
c     write(6,*) ' b(-) = ',( b(i),i=1,ndof)
c     print *, '*** passed numass ***'
c     print *, 'ad(-) = ',(ad(i),i=1,ndof)
c     print *, 'an(-) = ',(an(i),i=1,ncoef1)
c     print *, ' b(-) = ',( b(i),i=1,ndof)

c                 print *,'xx inside assembly end do 4= ',(xx(i),i=1,42)
c                 print *,'yy inside assembly end do 4= ',(yy(i),i=1,42)
c                 print *,'zz inside assembly end do 4= ',(zz(i),i=1,42)
c                 print *,'&&&&&&&&&& end do 4 &&&&&&&&&&&&&'
```

```
c                    pausc

chk                  If(icheck .eq. 1) then

chk     print *, 'ad diag stiff final = ',(ad(i),i=1,ndof)
chk     print *, 'an off diag stiff final = ',(an(i),i=1,ncoef1)

chk                   pause
chk                  Endif

 4    continue

 98   continue
c                    stop
c      print *, 'ad(-) = ',(ad(i),i=1,ndof)

c                    stop
c      write(6,*), 'ad diag stiff final = ',(ad(i),i=1,ndof)
c      write(6,*), 'an off diag stiff final = ',(an(i),i=1,ncoef1)
c      write(6,*),' b(-) at end assembly = ',( b(i),i=1,ndof)
c      print *, 'ad diag stiff final = ',(ad(i),i=1,ndof)
c                     pause

cc      print *, 'an off diag stiff final = ',(an(i),i=1,ncoef1)
c
c                    stop
      return
      end

c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine generalip(nel,ndofpe,neltype,maxdofpe,nboundc
    $,            ndofpn,numnodes,loadof,nummat,nlayers,nsect
    $,            lumpmass,ndof,ieall,jeall,ianal,neig,iarc
    $,            itype)

c
c if "iarc"=1 then use Arc Length method
c if "itype"=6 then 2d nonlinear truss element
c
c......Purpose: Input some general input information
c
      dimension nel(1),ndofpe(1)
c       common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)
c
      read(5,*) nboundc,ndofpn,numnodes,loadof,nummat
     $,nlayers,nsect,lumpmass,ianal,neig,iarc,itype,itruss
c
c......ndofpe(1,2,3,4,5,6)=(4,12,24,15,99,99)
c                   = 2-d truss
c                   = 3-d beam
c                   = 2-d BFS rectangular plate (bending+membrane)
c                   = 3-d MIN3 triangular plate
c                   =    "future" element type
c                   =    "future" element type
      read(5,*) (nel(i),i=1,neltype)
      read(5,*) (ndofpe(i),i=1,neltype)
      write(6,*) 'nboundc,ndofpn,numnodes,loadof'
      write(6,*) 'nummat,nlayers,nsect,lumpmass,ianal,neig'
      write(6,*)  nboundc,ndofpn,numnodes,loadof
      write(6,*)  nummat,nlayers,nsect,lumpmass,ianal,neig
      write(6,*) 'nel(neltype) = ',(nel(i),i=1,neltype)
      write(6,*) 'ndofpe(neltype) = ',(ndofpe(i),i=1,neltype)
      ndof=numnodes*ndofpn
c                    print *,'ndofpe in gen = ',ndofpe
c                    print *,'ndof in gen = ',ndof
c                    print *,'nel inside generalip=',nel
c                    pause
c                    stop
c
       maxdofpe=6      ! for 3-d truss
       ieall=0
       jeall=0
       do 1 i=1,neltype
       if (nel(i) .eq. 0) go to 1
        if (ndofpe(i) .gt. maxdofpe) maxdofpe=ndofpe(i)
        ieall=ieall+nel(i)
        jeall=jeall+nel(i)*ndofpe(i)
 1      continue
c                     print *,'end of generalip'
c
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine supernode(ndof,isupn)
      dimension isupn(1)
c
c      Purposes: Find super node information (for unrolling purpose)
c
      do 11 i=1,ndof
      isupn(i)=1
 11   continue
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine minfills(ndof,iperm)
      dimension iperm(1)
c
c      Purpose: Reordering algorithms, to minimize fill-in terms
c
      do 1 i=1,ndof
      iperm(i)=i
 1    continue
c                     print *,'iperm inside minfills=',(iperm(i),i=1,ndof)
c                     stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
      subroutine copy_int(n,iarray,icopy)
      dimension iarray(1),icopy(1)
c                    print *,'Check inside copy_int'
c                    print *,'n=',n
c                    print *,'iarray(1)=',iarray(1)
       do 2 i=1,n
       icopy(i)=iarray(i)
c                    print *,'icopy(i) inside copy_int=',icopy(i)
 2     continue
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
c    A R C   L E N G T H
c
c This subroutine added for Arc Length method.  Also
c updated elstif for 2D nonlinear truss.
c Rev 3/16/04
c
C*********************************************************************
                    Subroutine rikarc(ie,je,lm,neltype,nel,ndofpe,ndofpn,
     $  numnodes,x,y,z,
     $  ndof,b,loadof,
     $  iboundc,nboundc,ia,
c     $  iboundc,nboundc,ia1,ia2,
     $  ieall,ji,iet,jet,
     $  itempo1,exfor,
c     $  ja,
     $  ip,ad,an,propmat,propsect,be, !call assembly
c     $  ip,ad,an,propmat,propsect,elk,be, !call assembly
     $  ae,maxdofpe,
     $  ianal,lumpmass,dm,am,elm,tempo1,
     $  iperm,  !call minfills
     $  in,ncoef2,ju, !call symfactd
     $  iu,iut,  !call transad
     $  isupn, !call supernode
     $  di,un,iup,iopf,  !call numfa1d
     $  iopb,  !call fbed
     $  disp,iarc,jeall,ux,uy,
     $  qk,zee,dLength,iex)
c     $  ja,qk)
c     $  ja,esm,elk,qk)
C*********************************************************************
C***            PROGRAM: RIK2DOF
C***
C***    Program to trace the equilibrium path for a geometrically
C***    nonlinear single degree of freedom truss.
C***
C***    Input Variables:
C***     dl0 = starting incremental external load multiplier,
C**          delta lambda 0, (real, scalar)
C***     dlmax = maximum allowable incremental external load
C***            multiplier, delta lambda max, (real, scalar)
C***     dlmin = minimum allowable incremental external load
C***            multiplier, delta lambda min, (real, scalar)
C***     exfor = external force, Q, (real, vector)
C***     exlmx = maximum allowable total external load
C***            multiplier, lambda max, (real, scalar)
C***     itedes = number of desired iterations for
C***             convergence, (integer, scalar)
C***     itemax = maximum allowable number of iterations,
C***             (integer, scalar)
C***     mxstp = maximum allowable number of load steps,
C***            (integer, scalar)
C***     toldis = convergence tolerance for displacements,
C***             (real, scalar)
C***     tolfor = convergence tolerance for residual forces,
C***             (real, scalar)
C***
C***    Output Variables:
C***     exlk = total external load multiplier at step k (or i),
C***           lambda k (or lambda i), (real, scalar)
C***     qk = total displacements at step k (or i),
C***          qk (or qi), (real, vector)
C***
C***    Intermediate Variables:
C***     dli = incremental external load multiplier at step i,
C***          delta lambda i, (real, scalar)
C***     dlim1 = incremental external load multiplier at step i-1,
C***            delta lambda i-1, (real, scalar)
C***     dlk = incremental external load multiplier at step k,
C***          delta lambda k, (real, scalar)
C***     dqi = incremental displacements at step i, delta qi,
C***          (real, vector)
C***     dqim1 = incremental displacements at step i-1, delta qi-1,
C***            (real, vector)
C***     dqk = incremental displacements at step k, delta qk,
C***          (real, vector)
C***     dqk1 = part 1 of delta qk, delta qkI, (real, vector)
C***     dqk2 = part 2 of delta qk, delta qkII, (real, vector)
C***     dqtot = total displacements at load step i, delta qtot,
C***            (real, vector)
C***     dqsq = storage for dqtot*dqtot, (real, scalar)
C***     ds0 = magnitude of initial tangent vector, delta s0,
C***          (real, scalar)
C***     dsi = magnitude of the tangent vector at load step i,
C***          delta si, (real, scalar)
C***     dsim1 = magnitude of the tangent vector at load step i-1,
C***            delta si-1, (real scalar)
C***     dsmax = maximum allowable magnitude of the tangent
C***            vector, delta smax, (real, scalar)
C***     dsmin = minimum allowable magnitude of the tangent
C***            vector, delta smin, (real, scalar)
C***     fintk = internal force at load step k (or i),
C***            Fk (or Fi), (real, vector)
C***     iconv = convergence flag, iconv=0 means toldis and
C***            and tolfor have been met, (integer, scalar)
C***     iflag = error flag for iteration on the normal
C***            plane, iflag .ne. 0 means maximum iterations
C***            have been exceeded, (integer, scalar)
C***     istep = load step counter (integer, scalar)
C***     it = iteration counter (integer, scalar)
```

```
C***     resk = residual force at load step k (or i),
C***          FRk (or FRi), (real, vector)
C***     scali = scale factor for load step i, Si, (real, vector)
C***     stifi = tangent stiffness at i, KTi, (real, matrix)
C***     stifk = tangent stiffness at k, KTk, (real, matrix)
C***
C***
C**********************************************************************
*
      implicit real*8(a-h,o-z)
c     parameter (neq=2)

      common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c     common/deltaqi/dqi(500)

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
c     common/disp/ux(2),uy(2),uz(2),disp(500)
      common/mfor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/testing/toldis,tolfor,exlk,iconv
      common/nodeforc/ibstor(500),bvstor(500)

                    common/jam/ja2(100)

c     common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)

c     double precision, allocatable:: dqtot(:),dqi(:),dqim1(:),qk(:)
      double precision, allocatable:: dqtot(:),dqi(:),dqim1(:)
      integer, allocatable:: itempo2(:)
c     double precision, allocatable:: ja(:)

                    dimension disp(*),b(*),ia(*),iperm(*),qk(*)
c                   dimension disp(*),b(*),ia1(*),ia2(*),iperm(*)
                    dimension ie(*),je(*),iet(*),jet(*),ae(*),ad(*)
                    dimension iboundc(*),an(*),ux(*),uy(*),di(*),exfor(*)
                    dimension x(*),y(*),z(*),ip(*),lm(1),iu(*)
c                   dimension ja(*),x(*),y(*),z(*),ip(*),lm(1),iu(*)

c                   dimension elk(maxdofpe,maxdofpe)

c ... open files for output
c     open(15,file='fearc_out.txt',status='old')
c     open(12,file='fearc_out_F.txt',status='old')
c     open(13,file='fearc_out_q1.txt',status='old')

c     open(15,file='fea3d2_out.txt',status='old')
c     open(12,file='fea3d2_out_F.txt',status='old')
c     open(13,file='fea3d2_out_q1.txt',status='old')


c     open(14,file='fea3d2_out_q2.txt',status='old')

c                   allocate (ja(50))

                    neq=ndof
c                   print *,'numnodes=',numnodes
c                   print *,'x start rik =',(x(i),i=1,numnodes)
c                   print *,'y start rik =',(y(i),i=1,numnodes)
c                   print *,'z start rik =',(z(i),i=1,numnodes)
c                   stop
c                   print *,'b start rikarc = ',(b(i),i=1,ndof)
c                   stop
c     print *,'jgsm = ',jgsm
cc     print *,'np = ',np
c     print *,'neq inside Rikarc = ',neq
c     print *,'a at start of RIK= ',a
c     print *,'dqi at start of RIK=',dqi
c      print *,'start of RIK'
c                     print *,'ityp start RIK=',ityp
c                     print *,'ndof start RIK=',ndof
c                   stop
c                   print *,'maxdofpe=',maxdofpe
c       stop
c c     pause
c ... define user input
c** dl0 = starting incremental external load multiplier
c     dl0 = 2.5d-2
ccccc      dl0 = 1.d-2
c     dl0 = 1.d-1
c4       dl0 = 1.d0
cc       dl0 = 1.d-3
ccc        dl0 = .001
C***     dlmax = maximum allowable incremental external load
C***           multiplier, delta lambda max, (real, scalar)
C***     dlmin = minimum allowable incremental external load
C***           multiplier, delta lambda min, (real, scalar)
c     dlmax = 2.6d-2
ccccc      dlmax = 1.d-2
ccccc      dlmin = 1.d-3
ccc       dlmax = 1.d-3
ccc       dlmin = 1.d-4
cc       dlmax = 1.d-4
cc       dlmin = 1.d-5
ccccccc         dlmax = .1
ccccccc         dlmin = .01
ccccc        dlmax = 1.
ccccc        dlmin = .1
c4        dlmax = 1000.
c4        dlmin = 10.
cxx         dlmin = 1.
c** Use below for truss1
c     dlmax = 1.d-0
c     dlmin = 1.d-1
c** Use below for truss2
c     dlmax = 1.d+1
c     dlmin = 1.d-0
c** read in the intial forces from big "a"
cc     print *,'jgf = ',jgf
c     stop
cc     print *,'exfor at start of RikArc = ',exfor
c     stop
c     exfor(1) = 1.55d0
c     exfor(2) = 1.45d0
C***      exlmx = maximum allowable total external load
C***            multiplier, lambda max, (real, scalar)
ccccc      exlmx = 1.d0
cccc       exlmx = .5d0
cc        exlmx = .2d0
cc      exlmx = .1d0
c4      exlmx = 100.d0

c     itedes = 4
c      itedes = 1
c     itemax = 20000
c     itemax =999999
c     itemax = 1500
c     itemax = 900000

      itemax = 1000000

cc      mxstp = 60
czzz      mxstp = 1500
ccc       mxstp = 25000

       mxstp = 1000000 !istep

cccccc      toldis = 1.d-3
cccccc      tolfor = 1.d-3
cxx      toldis = 1.d-2
cxx      tolfor = 1.d-2
cccc      toldis = .5
cccc      tolfor = .5
c4      toldis = 1.d-1
c4      tolfor = 1.d-1
c       toldis = 1.5d0
c       tolfor = 1.5d0
C**********************************************************
c** Truss_2bay_3D.dat
C** Symmetric
C** This is 6 nodes, 11 element
c      itedes = 1
c       dl0 = .01d0
c       dlmax = .001d0
c        dlmin = .0001d0
c      exlmx = 1.d0
c     toldis = 1.d-6
c     tolfor = 1.d-6
C**********************************************************
C**********************************************************
c** Two Bar Truss
C** Symmetric
C** This is 3 nodes, 2 element
cc       itedes = 1
cc        dl0 = .01d0
cc        dlmax = .01d0
cc         dlmin = .001d0
cc       exlmx = 1.d0
cc      toldis = 1.d-6
cc      tolfor = 1.d-6
c                   print *,'Check After input'
c                   pause
C**********************************************************
C**********************************************************
c** Tripod
C** Crisfield 3.10.2
C** This is 4 nodes, 2 element
cpod       itedes = 1
cpod        dl0 = .1d0
cpod        dlmax = .001d0
cpod         dlmin = .0001d0
cpod       exlmx = 1.d0
cpod      toldis = 1.d-6
cpod      tolfor = 1.d-6
c      toldis = 1.d-12
c      tolfor = 1.d-12
C**********************************************************
C**********************************************************
C** Crisfield truss, ex 22.4a, vol2 chap 22.4, page 407
C** cris_22_4a_in.dat  6nodes, 9 elements
c       itedes=1
c       dl0 = .1d0
c       dlmax = .01d0
c        dlmin = .001d0
c      exlmx = 1.d0
c     toldis = 1.d-6
c     tolfor = 1.d-6
C**********************************************************
C**********************************************************
C** Duc truss, truss11_3d_in.dat
C** Duc truss, truss11_3d_xz_in.dat
C** This is a 6 nodes, 11 elements,
c       itedes=1
c       dl0 = .01d0
c       dlmax = .0001d0
c        dlmin = .00001d0
c      exlmx = 1.d0
c     toldis = 1.d-6
c     tolfor = 1.d-6
C**********************************************************

C**********************************************************
C** Duc truss, truss10_3d_in.dat
C** This is a 6 nodes, 10 elements,
c       itedes=1
c       dl0 = .01d0
c       dlmax = .001d0
c        dlmin = .0001d0
c      exlmx = 1.d0
c     toldis = 1.d-6
c     tolfor = 1.d-6
C**********************************************************
C** Duc truss, truss7_3d_in.dat
```

```
C** This is a 5 nodes, 7 elements,
ctruss       itedes=1
ctruss       dl0 = .01d0
ctruss       dlmax = .001d0
ctruss       dlmin = .0001d0
ctruss      exlmx = 1.d0
ctruss     toldis = 1.d-4
ctruss     tolfor = 1.d-4
C*************************************************************
C** The following are used for test problelm Truss3in.dat& T4in.dat
C** This is a 2 ele, 3 node problem, results agree with Crisfield
c       dl0 = .1d0
c       dlmax = .001
c       dlmin = .0001
c      exlmx = 1.d0
c     toldis = .00001
c     tolfor = .00001d0
C*************************************************************
C** The following are used for test problelm T3in.dat
C** This is a 3 ele, 3 node problem, results agree with Crisfield
c       dl0 = .1d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c     toldis = .01d0
c     tolfor = .01d0
C*************************************************************
C** The following are used for test problelm T3xin.dat & T3ain.dat
C** This is a 6 ele, 4 node problem, results agree with Crisfield
c       dl0 = .1d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c     toldis = .00001d0
c     tolfor = .00001d0
C*************************************************************
C** The following are used for test problelm T3xin.dat
C** This is a 6 ele, 4 node problem, results agree with nastran
c       dl0 = .1d0
c       dlmax = .1
c       dlmin = .01
c      exlmx = 1.d0
c     toldis = .000001d0
c     tolfor = .01d0
C*************************************************************
C** The following are used for test problelm  T4xxxin.dat GOOD
C** This is a 10 ele, 7 node problem, results agree with nastran
c       dl0 = .1d0
c       dlmax = .1d0
c       dlmin = .01d0
c      exlmx = 1.d0
c     toldis = 1.d-9
c     tolfor = 1.d-9
C*************************************************************
C** Crisfield small arch,  arch20in.dat
C** This is a 10 nodes, 20 element,
c       dl0 = .1d0
c       dlmax = .001d0
c       dlmin = .0001d0
c      exlmx = 1.d0
c     toldis = 1.d-7
c     tolfor = 1.d-7
C*************************************************************
C** Crisfield small arch,  arch18in.dat
C** This is a 18 nodes, 41 element,
c       dl0 = .01d0
c       dlmax = .001d0
c       dlmin = .0001d0
c      exlmx = 1.d0
c     toldis = 1.d-5
c     tolfor = 1.d-5
C*************************************************************
C*************************************************************
C** Crisfield small arch,  arch10_3D_in.dat
C** This is a 10 nodes, 21 element,
c        itedes=1
c       dl0 = .001d0
c       dlmax = .001d0
c       dlmin = .0001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*************************************************************
C** Crisfield small arch,  arch18_3D_in.dat
C** This is a 18 nodes, 41 element,
c        itedes=5
c       dl0 = .01d0
c       dlmax = .005d0
c       dlmin = .0005d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*************************************************************
C** Crisfield large symmetry arch,  arch22in.dat
C** This is a 22 nodes, 51 elements,
cc       dl0 = .1d0
cc       dlmax = .1d0
cc       dlmin = .01d0
cc      exlmx = 1.d0
cc     toldis = 1.d-3
cc     tolfor = 1.d-4
C*************************************************************
C** Crisfield small arch,  arch42_3D_in.dat
C**    arch3din.dat
C** This is a 42 nodes, 101 element,
c       itedes=1
c       dl0 = .1d0
c       dlmax = .01d0
c       dlmin = .0001d0
c      exlmx = 1.d0
c     toldis = 1.d-4
c     tolfor = 1.d-4
C*************************************************************
C** Crisfield small arch,  arch42in.dat
C**    arch10_3D_in.dat
C** This is a 21 nodes, 10 element,
carch                 itedes=1
carch       dl0 = .01d0
carch       dlmax = .01d0
carch       dlmin = .001d0
carch      exlmx = 1.d0
carch      toldis = 1.d-4
carch      tolfor = 1.d-4
C*************************************************************

C** Duc frame example  frame_in.dat
C**
C** This is a 4 nodes, 5 element,
c                 itedes=1
c       dl0 = .01d0
c       dlmax = .001d0
c       dlmin = .0001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6

c                 itedes=2
c       dl0 = .025d0
c       dlmax = 8.d-3
c       dlmin = 5.d-3
c      exlmx = 1.d0
c      toldis = 1.d-3
c      tolfor = 1.d-4
C*************************************************************

C*************************************************************
C**    t3a_in.dat
C** This is a 3 nodes, 3 element,
ct3a       itedes=1
ct3a       dl0 = .00001d0
ct3a       dlmax = 1.d-3
ct3a       dlmin = 1.d-4
ct3a      exlmx = 1.d0
ct3a      toldis = 1.d-6
ct3a      tolfor = 1.d-6
C*************************************************************

C** 3D Truss , Td3_in.dat
C** This is a 4 nodes, 3 elements,
c       dl0 = .1d0
c       dlmax = .01d0
c       dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-5
c      tolfor = 1.d-5
C*************************************************************
C** 3D Truss , Dome_in.dat & Dome2_in.dat
C** This is a 13 nodes, 24 elements,
cdome      itedes=1
cdome       dl0 = .1d0
cdome       dlmax = .1d0
cdome       dlmin = .001d0
cdome      exlmx = 1.d0
cdome      toldis = 1.d-6
cdome      tolfor = 1.d-6
C*************************************************************
C** Khot_shallow_truss
C**
ckhot                 itedes=1
ckhot       dl0 = .001d0
ckhot       dlmax = .001d0
ckhot       dlmin = .0001d0
ckhot      exlmx = 1.d0
ckhot      toldis = 1.d-4
ckhot      tolfor = 1.d-4
C*************************************************************
C** snap_in.dat
C** This is a 4 nodes, 3 elements,
csnap      itedes=1
csnap       dl0 = .01d0
c      dl0 = .5d0
csnap       dlmax = .1d0
csnap       dlmin = .00001d0
csnap      exlmx = 1.d0
csnap      toldis = 1.d-4
csnap      tolfor = 1.d-4

C*************************************************************
C** 3D Truss , twobar_in.dat !Similar to Crisfield 22.1
C** large bifurcation
C** This is a 3 nodes, 2 elements,
ctwo       itedes=1
ctwo       dl0 = .001d0
c      dl0 = .5d0
ctwo       dlmax = .001d0
ctwo       dlmin = .0001d0
ctwo      exlmx = 1.d0
ctwo      toldis = 1.d-6
ctwo      tolfor = 1.d-6

C*************************************************************
C** The following are used for test problelm T4xin.dat & T4xxin.dat & T4in.dat
C** This is a 10 ele, 7 node problem, results agree with nastran
c       dl0 = .1d0
c       dlmax = .01d0
c       dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*************************************************************
C** The following are used for test problelm T5xxxin.dat & T4xxin.dat
C** This is a 10 ele, 7 node problem, results agree with nastran
c       dl0 = .1d0
c       dlmax = .01d0
```

```
c       dlmin = .001d0
c      exlmx = 1.d0
c     toldis = 1.d-5
c     tolfor = 1.d-4
C*************************************************************
C** The following are used for test problelm T7in.dat
C** This is a 4 ele, 5 node problem, results agree with Crisfield
c       dl0 = .01d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c     toldis = .01d0
c     tolfor = .01d0
C*************************************************************
C*************************************************************
C** The following are used for test problelm T7Ain.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
C** using these varibles, this takes a long time 30+ min
c       dl0 = .1d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c     toldis = .01d0
c     tolfor = .01d0
C** The following are used for test problelm T7Bin.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
C** try using different varibles for faster
c       dl0 = 1.d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c     toldis = .01d0
c     tolfor = .01d0
C** The following are used for test problelm T7Cin.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
C** try using different varibles for faster
c       dl0 = 1.d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = .1d0
c     toldis = .01d0
c     tolfor = .01d0
C** The following are used for test problelm T7Din.dat
C** This is a 6 ele, 5 node, 3 support problem, results agree with Crisfield
c       dl0 = .01d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c     toldis = .001d0
c     tolfor = .001d0
C** The following are used for test problelm T7iEn.dat
C** This is same as T7D above but use 2 supports instead of 3
C** This is a 6 ele, 5 node, 2 support problem
C** Trying to get a solution, do I need right bracketing or is program wrong?
c       dl0 = .01d0
c       dlmax = .01d0
c       dlmin = .001d0
c      exlmx = 1.d0
c .9 works, try smaller
c     toldis = .9d0
c     tolfor = .9d0
C*************************************************************
C*************************************************************
C** The following are used for test problelm T11in.dat
C** This is a 4 ele, 5 node problem, results agree with Crisfield
c       dl0 = 1.d0
c       dlmax = 1.
c       dlmin = .1
c      exlmx = 100.d0
c     toldis = .99d0
c     tolfor = .99d0
C*************************************************************
c
C*************************************************************
C** The following are used for test problelm T11in.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
c       dl0 = 1.d0
c       dlmax = .1
c       dlmin = .01
c      exlmx = 1000.d0
c     toldis = .9d0
c     tolfor = .98d0
C*************************************************************
C*************************************************************
C** The following are used for test problelm T11in.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
c       dl0 = 1.d0
c       dlmax = .1
c       dlmin = .01
c      exlmx = 100.d0
c     toldis = .99d0
c     tolfor = .99d0
C*************************************************************
c ... end user input

C***Crisfield sdof
c       dl0 = 1.0d-4
c       dlmax = 1.0d-3
c       dlmin = 1.0d-4
c       exfor(1) = -100.d0
C***Crisfield sdof

c      exfor(1) = 1.55d0
c      exfor(2) = 1.45d0
c>>> Arora's exact solution={-.397975,0.540417} <<<
c       dl0 = 1.0d-5
c       dlmax = 1.0d-5
c       dlmin = 1.0d-6
c      exfor(1) =  0.d0
c      exfor(2) = -3.d0

c>>> Arora's nonlinear 2 ex={-1/3,-1/2} <<<              OK

cc       dl0 = 1.0d-1
cc       dlmax = 1.0d-1
cc       dlmin = 1.0d-2
cc      exlmx = 1.d0
cc      itedes = 2
c     itemax = 100000
c     mxstp = 100000
cc      toldis = 1.d-3
cc      tolfor = 1.d-4

C***J. Dennis's opt. book page 172
c       dl0 = 1.0d-6
c       dlmax = 1.0d-3
c       dlmin = 1.0d-4
c      exfor(1) =  3.d0
c      exfor(2) =  9.d0
c      exlmx = 1.d0
c      itedes = 2
c      itemax = 10000
c      mxstp = 10000
c      toldis = 1.d-3
c      tolfor = 1.d-4

c***Arora's nonlinear page 331 soln={0,0}
       dl0 = 1.0d-1
       dlmax = 1.0d-1
       dlmin = 1.0d-2
      exlmx = 1.d0
      itedes = 2
      toldis = 1.d-3
      tolfor = 1.d-4

c***J. Dennis's opt book
c       dl0 = 1.0d-9
c       dlmax = 1.0d-10
c       dlmin = 1.0d-11
c      exlmx = 1.d0
c      itedes = 2
c      toldis = 1.d-3
c      tolfor = 1.d-4
c      exfor(1) =  1.0d-10
c      exfor(2) =  1.0d-10

c***Hrinda 1          **OK**
cc       dl0 = 1.d-2
cc       dlmax = 1.d-3
cc       dlmin = 1.d-4
cc      exlmx = 1.d0
cc      itedes = 2
cc      toldis = 1.d-3
cc      tolfor = 1.d-4

c***Hrinda 2          **OK**
cc       dl0 = 1.d-10
cc       dlmax = 1.d-10
cc       dlmin = 1.d-11
cc      exlmx = 1.d0
cc      itedes = 2
cc      toldis = 1.d-4
cc      tolfor = 1.d-5
c***Hrinda 3          **OK**
cc       dl0 = 1.d-1
cc       dlmax = 1.d-2
cc       dlmin = 1.d-3
cc      exlmx = 1.d0
cc      itedes = 2
cc      toldis = 1.d-4
cc      tolfor = 1.d-5
c
c***POLY TEST                          OK
c       dl0 = 1.d-8
c       dlmax = 1.d-8
c       dlmin = 1.d-9
c      exlmx = 1.d0
c      itedes = 2
c      toldis = 1.d-3
c      tolfor = 1.d-4
c***Matlab ex
c       dl0 = 1.d-3
c       dlmax = 1.d-3
c       dlmin = 1.d-4
c      exlmx = 1.d0
c      itedes = 2
c      toldis = 1.d-3
c      tolfor = 1.d-4

c                       allocate (dqtot(ndof),dqi(ndof),dqim1(ndof),qk(ndof))
                        allocate (dqtot(ndof),dqi(ndof),dqim1(ndof))

                        ncoef1_guess=20000000
      allocate ( itempo2(ncoef1_guess) )

c                       print *,'check BB'
c                       pause

c ... initialize required variables
c
      dli = 0.d0
      dlim1 = 0.d0
      dqsq = 0.d0
      ds0 = 0.d0
      dsi = 0.d0
      dsim1 = 0.d0
      dsmax = 0.d0
      dsmin = 0.d0
      exlk = 0.d0
      iconv = 0
      istep = 1
      it = 1
c
      do 20 i=1,neq
c       do 10 j=1,neq
```

```
c                        print *,'chk before stifi'
c                        pause
c           stifi(j,i) = 0.d0
c 10     continue
c
c                     print *,'check zz'
c                     pausc

      dqi(i) = 0.d0
c                                              print *,'Check after dqi'
c                     pause
      dqim1(i) = 0.d0
      dqtot(i) = 0.d0
      fintk(i) = 0.d0
      qk(i) = 0.d0
      resk(i) = 0.d0
      scali(i) = 0.d0
      work1(i) = 0.d0
 20   continue

c     print *,'exfor = ',(exfor(ic),ic=1,ndof)
c                     pause
c                      print *,'iarc=',iarc
c                                              print *,'maxdofpe=',maxdofpe
c                     print *,'iboundc=',(iboundc(i),i=1,ndof)
c       stop
c
    iskip = 0 !When zero will go thru symfactd, 1 will skip symfactd

    itnrmlon = 0 !When zero not going thru "itnrml"
c                When one, inside "itnrml" telling syseqn & assembly
c
 999  if (exlk .le. exlmx .and. istep .le. mxstp) then

c                     If(istep.eq.910)Then
c                          itedes=1
c       dl0 = .01d0
c       dlmax = .000001d0
c       dlmin = .00000001d0
c       exlmx = 1.d0
c       toldis = 1.d-8
c       tolfor = 1.d-8
c                     print *,'Change starting arc varibles'
c                     Endif

c      write(*,*) 'Beginning Load Step Number in RIKARC:',istep
c                     print *,'exlk start Rik=',exlk
c       pause
c       stop
      it = 1
      if (istep .eq. 1) then
c     print *,'** CHECK A **'
c                     pause
c                     print *,'je start istep=1  = ',(je(i),i=1,20)

c      print *,'getstiff=',getstiff
c      print *,'getforce=',getforce
c      print *,'getsolve=',getsolve
c      pause

c                     stop
c ... compute Kt0 for eqn. 4.107
c                     print *,'b before syseqn = ',(b(i),i=1,ndof)
c                     pausc
c                     stop
c                     print *,'maxdofpe=',maxdofpe
c                     print *,'check before 1st syseqn'
c                     pause
c     stop
c                     getstiff=1

      call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $          ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $          neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $          iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $        ux,uy,uz,di,it,area,youngm,
     $      ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $ zee,dLength,iex)

c                                     write(6,*)'ia2 after syseqn=',(ia2(i),i=1,ndof+1)
c    $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c      print *,'chk after syseqn in rikarc'
c                     print *,'********************'
c                     print *,'********************'
c                     print *,'disp after 1st syseqn=',(disp(i),i=1,ndof)
c                     print *,'********************'
c                     print *,'********************'
c                     pause
c                     print *,'iskip=',iskip
c                     print *,'itempo2(1) after 1st syseqn=',itempo2(1)
c                     pause

c                      print *,'b after 1st syseqn=',(b(i),i=1,ndof)
c                     pause
c                     stop
c                     write(6,*),'disp after 1st syseqn=',(disp(i),i=1,ndof)
c                     write(6,*),'ae stiff after 1st syseqn=',(ae(i),i=1,maxdofpe**2)
c                     print *,'ad stiff after 1st syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                                     print *,'an stiff after 1st syseqn=',(an(ic),ic=1,50)
c                     print *,'ae stiff after 1st syseqn=',(ae(ic),ic=1,maxdofpe**2)
c                     pause
c                     stop

         do 704 i=1,neq
c             scali(i) = 1.d0/stifi(i,i)
c     c c     print *,'a(neq*2+i) = ',a(neq*2+i)
cc              scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
             scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c                     print *,'scali =',scali(i)
 704      continue
c                     stop
c     print *,'a after syseqn = ',a
c          stop
         getstiff = 0
         getforce = 0
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c      print *,'a = ',a
c      stop

c      stop

    do 44 i=1,ndof
44   dqtot(i)=disp(i)
c     print *,'dqtot(i) = ',(dqtot(i),i=1,ndof)
c                     pause
c                     write(6,*)'dqtot( )=',(dqtot(i),i=1,ndof)
c      stop
c ... compute dqtot**2 for eqn. 4.109
         dprd1 = 0.d0
         do 50 i=1,neq
           dprd1 = dprd1 + dqtot(i)**2
c     print *,'dprd1 = ',dprd1
 50       continue
c      write(6,*)'dprd=',dprd
cc     print *,'dprd1 = ',dprd1
c      stop
c ... compute ds0 using eqn. 4.109
       ds0 = dl0*dsqrt(dprd1+1.d0)
c ... compute dsmax using user specified max load inc
         dsmax = dlmax*dsqrt(dprd1+1.d0)
c ... compute dsmin using user specified min load inc
         dsmin = dlmin*dsqrt(dprd1+1.d0)
c ... compnte and store ds(i-1) for eqn. 4.110
         dsim1 = ds0

c     print *,'dsmax = ',dsmax
c     print *,'dsmin = ',dsmin
c     print *,'dsim1 = ',dsim1
c                     pause
c          stop

c ... compute dq0 using eqn. 4.106, and store in dqi
c ... store dq0 in dqim1 for next load step
c ... update the total displacements qk
c c     print *,'dl0 & dqtot(i) =',dl0,dqtot
c     print *,'neq = ',neq

c                      write(6,*)'qk(i) before=',(qk(i),i=1,neq)

c                     print *,'dl0=',dl0
c                     print *,'dqtot(_)=',(dqtot(ic),ic=1,neq)
c                     pause

         do 60 i=1,neq
           dqi(i) = dl0*dqtot(i)
c     print *,'dqi(i) = ',dqi
           dqim1(i) = dqi(i)
c     print *,'qk(i)before=',qk
c           disp(i) = disp(i) + dqi(i)
c                          print *,'dqi(i)=',dqi(i)
c                          print *,'disp(i) = disp(i) + dqi(i)',disp(i)
c                          pause
           qk(i) = qk(i) + dqi(i)
c     print *,'qk(i)after=',qk(i)
c                     pause
 60       continue
c                      write(6,*)'dl0=',dl0
c                      write(6,*)'dqtot(i)=',(dqtot(i),i=1,neq)
c                      write(6,*)'dqi(i)=',(dqi(i),i=1,neq)
c                      write(6,*)'qk(i)=',(qk(i),i=1,neq)

c          stop
c     print *,'dqtot(i) = ',dqtot
c     print *,'qk(i) = ',qk
c                     pause
c                     stop
c?? not sure ??
c** Put qk in "a"
ccc       Do 61 iqk = 1,neq
ccc       a(iqk) = qk(iqk)
ccc 61     continue

c** Put qk in "disp"
      Do 61 iqk = 1,neq
      disp(iqk) = qk(iqk)
 61    continue
c                     print *,'@@@@@@@@@@@@@@@@@@@@@@@'
c                     print *,'disp(i)=qk(k)',(disp(ic),ic=1,neq)
c                     print *,'@@@@@@@@@@@@@@@@@@@@@@@'
c                     pause

c     print *,'"a after insert qk" =',a
c       stop
c ... update the total external load multiplier totlk
         exlk = exlk + dl0
c                     print *,'exlk=',exlk
c ... store dl0 in dlim1 for next load step
         dlim1 = dl0
c                     print *,'dlim1=',dlim1
c                     pause
c ... compute the internal forces, fintk, at qk
         getforce = 1 !!!Yes get internal force
c     print *,'getforce = ',getforce
c                     write(6,*)'getforce=',getforce
c                     write(6,*)'exlk=',exlk
c                     write(6,*)'dl0=',dl0
c                      write(6,*)'b before 2nd syseqn=',(b(i),i=1,ndof)
c          stop
c*****************************************************
c        print *,' "a" before 2nd syseqn to getforce = ',a
c          stop
c                       print *,'it before 2nd syseqn in RIK=',it
```

```
c                    print *,'before assembly instead of call to syseqn'
c                    print *,'ja2(i) before assembly=',(ja2(ic),ic=1,10)
c                    print
*,'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&'
c                    pause
c                       call assembly(ip,ad,an,lm,ie,je,propmat,
c   $         propsect,elk,be,b,iboundc,
c   $         ae,ia,ja,x,y,z,ncoefl,maxdofpe,
c   $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c   $ tempo1,disp,iarc,it,area,youngm,esm)

c                   print *,'********************'
c                   print *,'********************'
c                   print *,'disp before 2nd syseqn=',(disp(i),i=1,ndof)
c                   print *,'********************'
c                   print *,'********************'

c                    print *,'b before 2nd syseqn=',(b(i),i=1,ndof)
c                    pause
       call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
   $            ieall,ie,jc,iet,jet,jeall,ia,ae,maxdofpc,
   $            neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
   $            iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
   $          ux,uy,uz,di,it,area,youngm,
   $        ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
   $ zee,dLength,iex)

c   $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c                    print *,'Chk after 2nd syseqn'
c                   pause
c                   print *,'********************'
c                   print *,'********************'
c                   print *,'disp after 2nd syseqn=',(disp(i),i=1,ndof)
c                   print *,'********************'
c                   print *,'********************'
c                   pause

c                   print *,'ad stiff after 2nd syseqn=',(ad(i),i=1,maxdofpe**2)
c                   pause
c                                 print *,'an stiff=',(an(i),i=1,30)
c                   print *,'ae stiff after 2nd syseqn=',(ae(i),i=1,maxdofpe**2)
c                   pause

c                   stop
c                    print *,'qk(-)after 2nd syseqn=',(qk(i),i=1,ndof)

c                    print *,'b after 2nd syseqn=',(b(i),i=1,ndof)
c                   pause
c
                     write(6,*)'b after 2nd syseqn=',(b(i),i=1,ndof)
c                    write(6,*)'Chk after 2nd syseqn'

c                    stop
c     print *,'Check after 2nd syseqn'
c                   stop
c     print *,'getforce = ',getforce
c     print *,'getstiff =',getstiff
c      print *,'a after getforce=',a
c      print *,'exlk after syseqn = ',exlk
c                    print *,'exfor before resid=',(exfor(i),i=1,neq)
c                    stop
ccc         do 705 i=1,neq
c             scali(i) = 1.d0/stifi(i,i)
c     c  c     print *,'a(neq*2+i) = ',a(neq*2+i)
cc               scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
ccc               scali(i) = 1.d0/ad(i)  !!!!Get diagonal

ccc                 print *,'scali =',scali(i)
ccc 705       continue

c                    stop
        getforce=0 !set back to zero
c*********************************************************
          call resid(dqtot,dqi,dqim1,qk,b,exfor)
c      print *,'resk after resid = ',resk
c      print *,'exlk after resid = ',exlk
c          stop
c ... test for convergence

c          call test(toldis,tolfor,exlk,iconv)
c                    print *,'Check before test'
c                   stop
        call test(dqtot,dqi,dqim1,qk,exfor,b,itnrmlon)
c      print *,'iconv after test = ',iconv
c      print *,'stop after test'
c      print *,'istep = ',istep
c      print *,'exlk = ',exlk
c                   pause
c          stop

        if (iconv .eq. 0) then
          call result(istep,it,dqtot,dqi,dqim1,qk,exfor,ad)
          istep = istep + 1

     print *,'stop after result and istep 1'
     pause
c            stop
          go to 999
        end if
c          istep = istep + 1
c     print *,'exlk right before else = ',exlk
c      print *,'a before else = ',a
c       stop
c))))))))))))))))))))))))))))))))))))))))))))))))
       else
c     print *,'** CHECK B **'
c                   print *,'istep in RIK=',istep
c                   pause
c     print *,'** CHECK B **'
c                   print *,'istep in RIK=',istep
c                   pause
c     print *,'** CHECK B **'
```

```
c                   print *,'istep in RIK=',istep
c                   pause
c                   print *,'ia(_)=',(ia(ic),ic=1,ndof)
c                   pause
c      stop
c))))))))))))))))))))))))))))))))))))))))))))))))
c ... compute dsi from eqn. 4.110
cc     print *,'after else '
        dsi = dsim1*dsqrt( dfloat(itedes)/dfloat(istep) )
c         dsi = dsim1*dsqrt( dfloat(itedes)/dfloat(it) )
c     print *,'Starting 2nd part of Rikarc'
c     print *,'dsi = ',dsi
c                   print *,'dsmax=',dsmax
c                   print *,'dsmin=',dsmin

c                   pause
c         stop
c     print *,'exlk after else =',exlk
c      stop
c ... test to see if dsi is between dsmax and dsmin
        if (dabs(dsi) .gt. dabs(dsmax)) then
          if (dsi .gt. 0.d0) then
            dsi = dsmax
c                   print *,'dsi check dsmax=',dsmax
          else
c         c     write(*,*) 'Program terminated!!'
c         c     write(*,*) 'Magnitude of tangent vector is negative'
            stop
          end if
        elseif (dabs(dsi) .lt. dabs(dsmin)) then
          if (dsi .gt. 0.d0) then
            dsi = dsmin
c                   print *,'dsi check dsmin=',dsi
          else
c         c     write(*,*) 'Program terminated!!'
c         c     write(*,*) 'Magnitude of tangent vector is negative'
            stop
          end if
        end if
c                   pause
c ... store dsim1 for next load step
        dsim1 = dsi
c ... compute Kti for eqn. 4.113

c     print *,'stop before syseqn'
c         stop


c         call stiff
c**Only get stiffness
       Getstiff=1
                    Getforce=0
                    Getsolve=0
c                   print *,'getstiff=',getstiff
c                   print *,'getforce=',getforce
c                   print *,'ja(_)=',(ja(ic),ic=1,3)
c                   pause
       call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
   $           icall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
   $           neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
   $           iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
   $           ux,uy,uz,di,it,area,youngm,
   $      ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
   $ zee,dLength,iex)

c    $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c**I call modify to get the stiffness w/delected rows/columns

c                   print *,'ia(_)=',(ia(ic),ic=1,ndof)
c                   pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                   pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                   pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                   pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                   pause
c      stop

cc       call modify
c     print *,'Check after third modify'
c     print *,'a after 3rd modify =',a
c     print *,'exlk after 3rd modify =',exlk
c      stop
c  c     pause
c c    print *,'ibstor = ',ibstor
c c    print *,'bvstor = ',bvstor
       Getstiff = 0 ! set getting just stiff back to zero
c                   print *,'getstiff=',getstiff
c                   print *,'getforce=',getforce
c                   print *,'getsolve=',getsolve
c                   pause
c      stop
c ... store scale factor for convergence testing, eqn. 4.133
c ... store scale factor for convergence testing, eqn. 4.133
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c     If(ityp .eq.  6)Then
c                   print *,'ad(_)=',(ad(i),i=1,neq)
c                   stop
       do 70 i=1,neq
c            scali(i) = 1.d0/stifi(i,i)
c     c  c     print *,'a(neq*2+i) = ',a(neq*2+i)
cc             scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
             scali(i) = 1.d0/ad(i)  !!!!Get diagonal
c              scali(i) = dabs(1.d0/ad(i))  !!!!Get diagonal
c                   print *,'scali =',scali(i)
 70       continue
c                   pause
c                   stop
c     Endif
c     If(ityp .eq.  12)Then
```

```
c          do 71 i=1,neq
c             scali(i) = 1.d0/stifi(i,i)
c     c c     print *,'a(neq*2+i) = ',a(neq*2+i)
c             scali(i) = 1.d0/a(neq*3+i)  !!!!Get diagonal
c 71       continue
c       Endif
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

c     print *,'scali(i) check = ',(scali(kk),i=1,neq)
c          stop
c          do 70 i=1,neq
c             scali(i) = 1.d0/stifi(i,i)
c 70       continue
c ... solve eqn. 4.113 for dqtot
           do 80 i=1,neq
             dqtot(i) = exfor(i)
 80        continue

c       print *,'exfor(_)=',(exfor(ic),ic=1,ndof)
c                    pause
c**Only solve
        Getstiff=0
                     Getforce=0
                     Getsolve=1
      call sysoqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $             ieall,ie,je,iet,jet,jeall,ia,ac,maxdofpe,
     $             neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $             iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $             ux,uy,uz,di,it,area,youngm,
     $       ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $ zcc,dLength,iex)

                     Getsolve=0

                     Do 82 i=1,neq
                     dqtot(i)=disp(i) !get new disp(_) & set equal to dqtot(_)
82                   Continue

c       print *,'dqtot(_)= ',(dqtot(ic),ic=1,neq)
c                    pause
c       stop
c put dqtot(i) into "a(i)" in place of force
          jay=0
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c       If(ityp .eq. 6)Then
c          do 81 i=neq+1,neq*2
           do 81 i=1,neq
              jay=jay+1
c            a(i)=dqtot(jay)
             b(i)=dqtot(jay)
c                    print *,'b(i)=',b(i)
 81        continue
c                    pause
c       Endif
c       If(ityp .eq. 12)Then
c          do 82 i=neq+1,neq*3
c             jay=jay+1
c            a(i)=dqtot(jay)
c 82       continue
c       Endif
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
cc      print *,'a w/dqtot=',a
c          stop
c          call solve(dqtot)
cc  c     print *,'stifi before ludcmp = ',stifi
c          call ludcmp(stifi,n,np,indx,x)
cc  c     print *,'stifi after ludcmp = ',stifi
c           stop
c          call lubksb(stifi,n,np,indx,dqtot)
cc  c     print *,'stifi after ludcmp = ',stifi

c       write(*,*) 'Before dcmpbd inside rikarc'
c       stop
        ipick = 3
cc      call dcmpbd
c       stop
c       write(*,*) 'After dcmpbd inside rikarc'
c       stop
cc      call slvbd
c       write(*,*) 'After slvbd & ipick=3 inside rikarc'
cc      print *,'a after slvbd=',a
c       stop

c ... compute dqtot**2 for eqn. 4.109
          dprd1 = 0.d0
          do 90 i=1,neq
            dprd1 = dprd1 + dqtot(i)**2
c           dprd1 = dprd1 + a(i)**2
 90       continue
c ... dli using eqn. 4.112
          dli = dsi/dsqrt(1.d0 + dprd1)
c                    If(dprd1.lt.0.)Then
c         print *,'dli inside RIK=',dli
c                    stop
c                    Endif
c         pause
c         stop
c ... test for sign of dli using eqn. 4.117
          dprd2 = 0.d0
          do 100 i=1,neq
            dprd2 = dprd2 + dqtot(i)*dqim1(i)
c           dprd2 = dprd2 + a(i)*dqim1(i)
c           dprd2 = dprd2 + b(i)*dqim1(i)
 100      continue
c                    print *,'istep=',istep
c                    print *,'dqtot(_)=',(dqtot(ic),ic=1,neq)
c                    print *,'dqim(_)=',(dqim1(ic),ic=1,neq)
c                    pause

                     If(dprd2.lt.0.)Then
c         print *,'dprd2 negative=',dprd2
c                     pause
c                    stop
                     Endif

                     If(dlim1.lt.0.)Then
c                    print *,'istep=',istep
c     print *,'dlim1 negative=',dlim1
c                    pause
c                    stop
                     Endif

c     print *,'dprd1=',dprd1
c     print *,'dprd2=',dprd2
c                    pause
c       stop
          testv = dli*(dprd2 + dlim1)
c                    print *,'testv=',testv
c                    print *,'istep=',istep
c                    pause



cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          if (testv .lt. 0.D0) then
c           if (testv .lc. 0.D0) then
            dli = -1.d0*dli
c             dli = 1.d0*dli
c             print *,'test for negative sign'
c                      pause
c           stop
cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          else
            dli = 1.d0*dli
c             dli = -1.d0*dli
c             print *,'test for positive sign'
c             stop
          end if

c                                    If(istep .eq. 914)then
c                    print *,'it=',it
c                    pause
c                                    If(it .gt. 4)then
c                                    If(ad(6) .lt. 0.)then
c                                    dli = -1.d0*dli
c                       exlk=-exlk
c                                    Endif

c     pause

c ... compute dqi using eqn. 4.114
c ... store dqi in dqim1 for next load step
c ... update the total displacements qk
          do 112 i=1,neq
            dqi(i) = dli*dqtot(i)
c           dqi(i) = dli*a(i)
c           dqi(i) = dli*b(i)
            dqim1(i) = dqi(i)
            qk(i) = qk(i) + dqi(i)

c** Need to put qk into "a" at displacement locations
c           a(i) = qk(i)
c                       b(i) = qk(i)
                        dqtot(i)=qk(i)
 112       continue
c     print *,'qk(i) = ',qk
cc     print *,'a w/qk update added =',a

c       stop
c ... update the total external load multiplier exlk
c     print *,'exlk before update =',exlk
c       stop
c     If(istep .eq. 561) then
c                    dli=abs(dli)
c                    endif
          exlk = exlk + dli

c                    If(istep .eq. 561) then
c                     exlk=-exlk
c                    endif
c     print *,'dli = ',dli
c     print *,'exlk after update =',exlk
c          stop
c c     print *,'exlk BBBBBBBB = ',exlk
c          stop
c ... store dli in dlim1 for next load step
          dlim1 = dli
c ... compute the internal forces, fintk, at qk
c           call forces
            getforce = 1 !!!Yes get internal force
c c     print *,'getforce = ',getforce
c           stop
c**********************************************************
c c     print *,' "a" before 2nd sysoqn to getforce = ',a
c                    print *,'ia(_)=',(ia(ic),ic=1,ndof)
c                    pause
c     print *,'********** after 2nd sysoqn after Else in Rikarc ******'
c                    pause
c     print *,'********** after 2nd sysoqn after Else in Rikarc ******'
c                    pause
c     print *,'********** after 2nd sysoqn after Else in Rikarc ******'
c                    pause
c     print *,'********** after 2nd sysoqn after Else in Rikarc ******'
c                    pause
c       stop
c          stop
        call sysoqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $             ieall,ie,je,iet,jet,jeall,ia,ac,maxdofpe,
     $             neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $             iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $             ux,uy,uz,di,it,area,youngm,
     $       ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $ zcc,dLength,iex)
```

```
c    $ja,iboundc,ndofped,bc,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)

c c     print *,'Check after 2nd syseqn'

c^^^^^^ SETTING ZERO DIAGONAL VALUES TO ONE ^^^^^^^^^^^^^^^^
c       Do 1013 i=neq*2+1,neq*3
                        Do 1013 i=1,neq
c       If(a(i) .GT. .01) GoTo 1013
cc        If(ad(i) .GT. .000001) GoTo 1013
        If(ad(i) .GT. 1.e-8) GoTo 1013

c         a(i)=1.0
c                       ad(i)=1.e-8
c                       ad(i)=0.0
1013    Continue
        do 701 i=1,neq
c               scali(i) = 1.d0/stifi(i,i)
c     c c       print *,'a(neq*2+i) = ',a(neq*2+i)
cc                scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
               scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c                       print *,'scali =',scali(i)
 701        continue

c c     print *,'getforce = ',getforce
c     print *,'a=',a
c         stop
cc          call modify
c     print *,'Check after forth modify'
c c     print *,' "a" after 4th modify = ',a
c       c     pause
          getforce=0 !set back to zero
c c     print *,' "a" after getforce = ',a
c       stop
c      print *,'Before 2nd resid'
c                       Stop
c ... compute the residual forces, resk, at qk and exlk
          call resid(dqtot,dqi,dqim1,qk,b,exfor)
c     print *,'exlk after resid = ',exlk
c     print *,'After resid'
c                       pause
c         stop
c ... test for convergence
c          call test(toldis,tolfor,exlk,iconv)
          call test(dqtot,dqi,dqim1,qk,exfor,b,itnrmlon)
c     print *,'After test'
c                       print *,'iconv after test = ',iconv
c      print *,'istep = ',istep
c                       pause
c          stop
          if (iconv .eq. 0) then
c     print *,'exlk before result=',exlk

c           stop
            call result(istep,it,dqtot,dqi,dqim1,qk,exfor,ad)

c                        print *,'after 2nd "call result" in rikarc'
c                       pause
           istep = istep + 1
c  c  c     print *,'istep = ',istep
c     print *,'Ready to go back to 999'
c     pause
c           stop
            go to 999
          end if
c)))))))))))))))))))))))))))))))))))))))))))))))))
        end if
c)))))))))))))))))))))))))))))))))))))))))))))))))
cc      print *,'"a" before itntml = ',a
c      print *,'ready to go into itnrml'
c     pause
c      print *,'ready to go into itnrml'
c     pause
c      print *,'ready to go into itnrml'
c     pause

c       stop
c
c       call itnrml(itemax,toldis,tolfor,dli,dlim1,
c    $                  exlk,iconv,iflag,it,nc,iplvl)
c     print *,'exlk right before itnrml = ',exlk
c       stop
c     print *,'exlk right before itnrml = ',exlk
c     print *,'dli before itnrml = ',dli
c     print *,'dqi before itnrml = ',dqi
c     print *,'a before itnrml = ',a
c      print *,'exfor before itnrml=',exfor
c       stop
c                       pause
c                       print *,'********************'
c                       print *,'********************'
c                       print *,'disp before itnrml=',(disp(i),i=1,ndof)
c                       print *,'********************'
c                       print *,'********************'
c     print *,'exfor before itnrml= ',(exfor(ic),ic=1,ndof)
c                       pause

c                       print *,'ad stiff before itnrml=',(ad(ic),ic=1,maxdofpe**2)
c                       pause
c                       print *,'an stiff before itnrml=',(an(ic),ic=1,50)
c                       print *,'b(_) before itnrml=',(b(ic),ic=1,ndof)
c                       pause
c                       print *,'ae(_)before itnrml=',(ae(ic),ic=1,18)
c                       pause

c                       pause
c     print *,'ia(-) before itnrml = ',(ia(i),i=1,ndof+1)
c                       PAUSE
c                       stop
        call itnrml(itemax,dli,dlim1,b,exfor,ad,neltype,qk,
     $          iflag,it,nc,iplvl,ndof,ia,nel,ncoefl,jeall,
     $            ieall,ic,jc,ict,jet,maxdofpe,dqtot,dqi,dqim1,
     $            ja,an,propmat,propsect,x,y,z,ndofpn,numnodes,
     $            ip,iu,disp,ianal,ndofpe,iskip,itempo2,iarc,lm,
     $            itnrmlon,iboundc,ae,di,istep,iex)

                        itnrmlon=0 !set back to zero because we are done going thru "itnrml"

c     print *,'exfor after itnrml= ',(exfor(ic),ic=1,ndof)
c                       pause
c     print *,'Check after itnrml'

c     print *,'exlk after itnrml = ',exlk
c       stop
cc      print *,'iflag = ',iflag
c     print *,'exlk right after itnrml = ',exlk
c     print *,'exfor(3) right after itnrml = ',exfor(3)
c     print *,'exfor(4) right after itnrml = ',exfor(4)
cc      print *,'a(10) right after itnrml = ',a(10)
c       stop
c     print *,'a after itnrml=',a
cc      print *,'ipick after itnrml=',ipick
c       stop
c
c     print *,'iconv call to itnrml = ',iconv
c     print *,'istep = ',istep
c     print *,'exlk = ',exlk
c                       pause
        if (iflag .eq. 0) then
c       print *,'Check before result'
c         call result(istep,it,dqtot,dqi,dqim1,qk,exfor)
          call result(istep,it,disp,dqi,dqim1,qk,exfor,ad)

c                        print *,'after 3rd "call result" in rikarc'
c                       pause

c                           print *,'qk(_)=',(qk(ic),ic=1,ndof)
c                           print *,'disp(_)=',(disp(ic),ic=1,ndof)
c                           pause

                        if(istep .eq. 255)then !write out stiffness diagonal
c                       if(istep .eq. 571)then !write out stiffness diagonal
c                        print *,'ad( ) stiff diag.',(ad(ic),ic=1,ndof)
c                       pause
                        endif
c       print *,'STOP3 after result'
c                           pause
c         stop
        elseif (iflag .eq. 2) then
          write(6,*) 'MAX. ITERATIONS EXCEEDED IN ITNRML!!!!'
          write(6,*) 'PROGRAM TERMINATED !!!!!'
c     c     write(*,*) 'MAX. ITERATIONS EXCEEDED IN ITNRML!!!!'
c     c     write(*,*) 'PROGRAM TERMINATED !!!!!'
          go to 5000
        else
          write(6,*) 'UNKNOWN ERROR IN ITNRML!!!!'
          write(6,*) 'PROGRAM TERMINATING!!!!'
          go to 5000
        end if
c
cccc            if (istep .GE. 7) stop

        istep = istep + 1
c
c     print *,'Ready to go to 999'
c       stop
        go to 999
c
      end if
c
c     print *,'Before 5000 stop'
c     print *,'exlk before stop=',exlk
c     print *,'cxlmx before stop=',cxlmx
c     print *,'istep before stop=',istep
c     print *,'mxstp before stop=',mxstp
 5000  stop
      end
C*********************************************************************

c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine syseqn(nc,iplvl,b,ndof,disp,iarc,qk,
     $          ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
c    $           ieall,ie,je,iet,jet,jeall,ia1,ia2,ae,maxdofpe,
     $          neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $          iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
c    $           iboundc,ad,elk,iperm,iu,ip,iut,dqtot,dqi,dqim1,
c    $           ux,uy,uz,di,it,area,youngm,esm,elk,
     $          ux,uy,uz,di,it,area,youngm,
     $          ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $ zce,dLength,icx)

c       call numfa1d(ndof,ia,ja,ad,an,iu,
c     $             ju,di,un,ip,iup,
c     $             isupn,iopf)

c     $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c**********
c,
c   generation of the system of equations
c
c**********
      implicit real*8(a-h,o-z)

can                     SAVE ja,an,itempo1
                        SAVE ja,itempo1
cccc      common/elmatx/esm(6,6),ef(6),x(6),y(6),z(6),kl,ndofpe
      common/element/ dx(5),dy(5),g(5),q(5),grdc(5,2)
      common/matl/dxc,dyc,ge,qe
      common/hcv/idbc(500,2),dbc(500,2),ndbc
      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
c     common/nodes/nel(500,4),nmtl(500),xc(500),yc(500),zc(500)
```

```
      common/nodes/nmtl(500),xc(500),yc(500),zc(500)
c     dimension dx(5),dy(5),g(5),q(5),grdc(5,2)

c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $        dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $        resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $        dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $        resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
     $  dqk2(5000),fintk(5000),
     $  resk(5000),scali(5000),work1(5000),indx(5000),neq

c        common/deltaqi/dqi(500)

c     common/disp/ux(2),uy(2),uz(2),disp(500)

c     common/disp/ux(2),uy(2),disp(10)
c     common/disp/ux(2),uy(2)
      common ns(6),phi(6),gdx(6),gdn(6),gdy(6)
c     common/disp/qk(500)
      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
c     common/eletype/ityp  !ityp = 6 for 2-D problems
      common/spring/springK
      common/testing/toldis,tolfor,exlk,iconv

                  common/jam/ja2(100)

c                 integer, allocatable:: ja(:),itempo1(:),itempo2(:)
                  integer, allocatable:: ja(:),itempo1(:),ju(:),jut(:)
c                 integer, allocatable:: itempo1(:)

c                 integer, allocatable:: itempo1(:)
ccc     double precision, allocatable:: an(:)

c     double precision, allocatable:: tempo1(:),an(:),ju(:),jut(:),un(:)
      double precision, allocatable:: tempo1(:),un(:)
c     double precision, allocatable:: ju(:),jut(:),un(:)
                   dimension b(*),disp(*),ux(*),uy(*),uz(*),x(*),y(*),z(*)
c                  dimension ie(*),je(*),iet(*),jet(*),ia(*)
can                dimension ie(*),je(*),iet(*),jet(*),ae(*),ad(*)
                   dimension ie(*),je(*),iet(*),jet(*),ae(*),ad(*),an(*)
                   dimension iu(*),ip(*),iut(*),iperm(*)
                   dimension dqtot(*),dqi(*),dqim1(*),di(*),ia(*),qk(*)
c                  dimension dqtot(*),dqi(*),dqim1(*),qk(*),di(*),ia1(*),ia2(*)
c                  dimension iboundc(*),elk(maxdofpe,1),ja(*),an(*),lm(1)
                   dimension iboundc(*),lm(1)
c                  dimension iboundc(*),elk(maxdofpe,maxdofpe),lm(1)
                  dimension itempo2(*),exfor(*)
c                  dimension ja(*)
c     $ja,iboundc,ndofped,bc,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c                  dimension be(*),ad(*),dm(*),am(*)

c                 dimension ja(19)

c                 dimension tempo1(10)
c                 print *,'S T A R T  O F  S Y S E Q N'
c                 pause
c                 print *,'iskip at start syseqn=',iskip
c                 print *,'getstiff at start syseqn=',getstiff
c                 print *,'getforce at start sysean=',getforce
c                 print *,'getsolve at start syseqn=',getsolve
c                 pause
c                 print *,'qk(_) at start of syseqn=',(qk(ic),ic=1,neq)
c                 pause

c                 print *,'b(_) inside start syseqn=',(b(ic),ic=1,neq)
c                 pause
c                 print *,'disp(_) inside start syseqn=',(disp(ic),ic=1,neq)
c                 pause
c                 print *,'ad(_) inside start syseqn=',(ad(ic),ic=1,neq)
c                 pause
c                 print *,'iskip start syseqn=',iskip
c                 pause

c...Solver needed, no update to stiff or forces required
c                 print *,'getsolve start syseqn=',getsolve
c                 pause
                  If(getsolve .eq. 1)Then
                  Goto 1111
                  Endif

                  If(getforce .eq. 1)then
c                 print *,'getforce =1 start syseqn, goto 32'
c                 pause
       Do 28 i=1,ndof
 28    b(i)=0.
                  goto 32
                  endif

                  allocate (tempo1(ndof))

c                 print *,'itnrmlon start syseqn=',itnrmlon
c                 pause
c*****************************
c*                If(itnrmlon .eq. 1)then
c*                do 27 i=1,neq
c*                dqk1(i) = exfor(i)
c*                dqk2(i) = resk(i)
c*                print *,'dqk1(_)=',dqk1(i)
c*                print *,'dqk2(_)=',dqk2(i)
c*27   continue
c*                j=0 !????????
c*                do 29 i=1,neq
c*                b(i)=dqk1(i)
c*                print *,'b(_) in syseqn & itnrmlon=',b(i)
c*29              continue
c*                Endif
c*****************************
c                 pause

c                 print *,'ianal at start of syseqn =',ianal
```

```
c                 print *,'lumpmass at start of syseqn =',lumpmass
c                 stop

c                 print *,'x start syseqn =',(x(i),i=1,ndof)
c                 print *,'y start syseqn =',(y(i),i=1,ndof)
c                 print *,'z start syseqn =',(z(i),i=1,ndof)
c                 print *,'ndof=',ndof
c                 stop
c                 print *,'chk start of syseqn'
c                 pause
c                 print *,'iarc=',iarc
c                 print *,'iboundc=',(iboundc(i),i=1,ndof)
c                 print *,'it = ',it
c                 stop

c                 print *,'maxdofpe =',maxdofpe
c                 stop
c                 print *,'b start of syseqn = ',(b(i),i=1,ndof)
c                 pause
c                 print *,'je start of syseqn = ',(je(i),i=1,20)
c                 stop

c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 stop
c                 endif

c                 print *,'getforce start syseqn=',getforce
c                 print *,'getstiff start syseqn=',getstiff
c                 stop
c                 pause

c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 stop
c                 endif

c       If(getforce .eq. 1)then
c     print *,'getforce=1 at start of syseqn'
c                 stop
c** Get ready for internal forces, set all forces to zero
c**************************************************
c       If(ityp .eq. 6)Then
c       Do 28 i=1,ndof
c 28    b(i)=0.
c**************************************************
c        stop
c       goto 30
c       endif

c** Get ready for new stiffness terms,
c** set only stiff & force terms to zero
c**************************************************
c       DO 29 i=1,ndof
c  29   b(i)=0.  !set loads to zero
c                 print *,'b = ',(b(i),i=1,ndof)
c                 stop
c                 ndof2=ndof**2
c                 print *,'ndof2=',ndof2
c                   Do 31 i=1,ndof2-10
c  31   an(i)=0.  !set stiff to zero
c                 print *,'chk 2 inside syseqn'
c                 stop

30                continue !just a blank, no Do associated
c......input boundary dof
c      call supportdof(iboundc,nboundc,b,ndof,ia)
c                 print *,'b inside syseqn = ',(b(i),i=1,ndof)
c                 stop
c......
cvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
                  IF(getforce .eq. 1)then  !skip transa2d, symbass, copy_int
c "Go to assembly2"
c                 print *,'ja(1)=',ja(1)
c                 print *,'getforce goto 32'
c                 pause
c                 stop
c                 print *,'ieall=',ieall
c                 print *,'ndof=',ndof
c                 print *,'jet = ',(jet(i),i=1,ieall*4)
c                 print *,'iet = ',(iet(i),i=1,ndof+1)
c     print *,'ie = ',(ie(i),i=1,ieall+1)
c                 print *,'je = ',(je(i),i=1,jeall)
c                 stop
                   goto 32
                  Endif
cvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 stop
c                 endif
c                 print *,'chk before transa2d'
c                 print *,'getstiff=',getstiff
c                 print *,'getforce=',getforce
c                 pause
c                 print *,'je inside syseqn before transa2d= ',(je(i),i=1,20)
c                 pause
c                 stop
c                 print *,'chk before transa2d'
c                 pause
      call transa2d(ieall,ndof,ie,je,iet,jet)

cc32    continue !dummy line from "if(getforce .eq. 1) above
c      write(6,*) '*** passed transa2 ***'
c                 print *,'after transa2d'
c                 pause
c                 stop
c                 print *,'ndof=',ndof
c                 print *,'jeall=',jeall
c                 print *,'ieall = ',ieall
c     print *,'ie = ',(ie(i),i=1,ieall+1)
c                 print *,'je = ',(je(i),i=1,jeall)
```

```
c                  print *,'je = ',(je(i),i=1,ndof*4)
c                  print *,'iet = ',(iet(i),i=1,ndof+1)
c                  print *,'jet = ',(jet(i),i=1,jeall)
c                  print *,'jet = ',(jet(i),i=1,ieall*4)
c                  print *,'ia1 = ',(ia1(i),i=1,ndof+1)
c                  stop
c                  if(it .eq. 2)then
c                  print *,'it =2'
c                  stop
c                  endif

c                  If(getforce .eq. 1)then
c                  print *,'getforce =1 before symbass'
c                  print *,'ndof=',ndof
c                  stop
c                  endif
c......sparse "symbolic" assembly
      ncoef1_guess=20000000
c                   deallocate (itempo1)
c
      allocate ( itempo1(ncoef1_guess) )
c                  print *,'iskip before allocate itempo2=',iskip
c                  pause
c                   If(iskip .eq. 1)Then
c                     itempo1(1)=itempo2(1)
c        print *,'itempo1(1)=itempo2(1)=',itempo1(1)
c                   Else
c                   Endif
c
c                   print *,'ndof=',ndof
c                   stop
c                  print *,'chk before symbass'
c                  print *,'getstiff=',getstiff
c                  print *,'getforce=',getforce
c                  print *,'ieall before symbass=',ieall
c                  print *,'jeall before symbass=',jeall
c                  print *,'ie before symbass=',(ie(ic),ic=1,ieall+1)
c                  print *,'je inside syseqn before symbass=',(je(i),i=1,30)
c                  print *,'iet before symbass=',(iet(ic),ic=1,ieall+1)
c                  print *,'jet before symbass=',(jet(ic),ic=1,ieall+1)
c                  print *,'ia before symbass=',(ia(i),i=1,ieall+1)
c                  print *,'ja before symbass=',(ja(i),i=1,ieall+1)
c                  stop
c                  pause
c                  print *,'iskip before if = ',iskip
c                  pause
                   If(iskip .eq. 1)Then !only do symbass one time, skip after 1st time
                   Go To 799
                   Else
      call symbass(ie,je,iet,jet,ndof,ia,
     $          itempo1,ncoef1)
c                   iskip = 1!This causes symbass to be skipped, only need to go once
                   itempo2(1)=itempo1(1) !Need this because itempo1 gets re-allocated
c       call symbass(ie,je,iet,jet,ndof,ia1,ia2,
c      $          itempo1,ncoef1)
      Endif
799   CONTINUE !dummy line for abv "If" statement
c                  print *,'iskip after symbass = ',iskip
c                  print *,'*** After symbass ***'
c                  print *,'je after syseqn = ',(je(i),i=1,20)
c                  print *,'itempo1=',itempo1(1)
c                  print *,'itempo2=',itempo2(1)
c                  pause

c                  stop
c                  pause

c                  If(getforce .eq. 1)then
c                  print *,'getforce =1 before symbass'
c                  print *,'ncoef1 =',ncoef1
c                  stop
c                  endif

c                  print *,'ncoef1 =',ncoef1
c                  pause
c                  stop
c                  print *,'ieall = ',ieall
c      print *,'ie = ',(ie(i),i=1,ieall+1)
c                  print *,'je = ',(je(i),i=1,jeall)
c                  print *,'iet = ',(iet(i),i=1,ndof+1)
c                  stop
c                  print *,'jeall=',jeall
c                  print *,'ndof=',ndof
c                  print *,'ncoef1=',ncoef1
c                  print *,'jet after symbass= ',(jet(i),i=1,jeall)
c                  print *,'ia1 after symbass= ',(ia1(i),i=1,ndof+1)
c                  stop
c                  print *,'ia2 after symbass= ',(ia2(i),i=1,ndof+1)
c                  print *,'itempo1 after symbass= ',(itempo1(i),i=1,ncoef1)
c                  pause
c                  stop

c                  If(getforce .eq. 1)then
c                  stop
c                  endif

c                  if(it .eq. 2)then
c                  print *,'it =2'
c                  print *,'STOP'
c                  stop
c                  endif

c                  stop
c                  print *,'ncoef1=',ncoef1
c                  stop
c                  do 111 i=1,ncoef1_guess
c111               print *,'itempo1 = ',itempo1(ncoef1_guess)
c                  stop
c                  print *,'ncoef1 = ',ncoef1
c      stop
c
c       allocate ( ja2(ncoef1) )
c                  stop
c...not sure, "ja" is also needed in assembly2
c                  print *,' ??????????????????????????????'
c...itnrmlon means call was done in subroutine itnrml
c                  print *,'itnrmlon=',itnrmlon
c                  pause
                   If(itnrmlon .eq. 1)Then !skip allocation, already done
                   goto 790
                   Endif
                   If(getstiff .eq. 0)then
c                   print *,'ncoef1=',ncoef1
c                  pause
      allocate ( ja(ncoef1) )
c                  print *,'allocate -ja- when getstiff=0 inside syseqn'
c                  pause
c      allocate ( ja2(ncoef1) )
can        allocate ( an(ncoef1) )
c                  print *,'allocate -an- when getstiff=0 inside syseqn'
c                  pause
                   Endif
790                Continue !dummy line

c                   allocate (tempo1(ndof))
c                  print *,'***chk before copy_int***'
c                  pause
c                  print *,'getstiff=',getstiff
c                  print *,'getforce=',getforce
c                  print *,'getsolve=',getsolve
c                  print *,'ncoef1=',ncoef1
c                  print *,'itempo1=',itempo1
c                  print *,'ja=',(ja(ic),ic=1,ncoef1)
c                  pause
c                  print *,'iskip before 1st copy_int=',iskip
c                  pause

                   If(iskip .eq. 1)Then !only do symbass one time, skip after 1st time
      goto 798
                   Else
      call copy_int(ncoef1,itempo1,ja)
     Endif
798   CONTINUE !dummy line for abv "If" statement
c                  print *,'passed 1st copy_int'
c                  print *,'ja(_) after copy_int in syseqn=',(ja(ic),ic=1,ncoef1)
c                  print *,'##############################'
c                  pause
c                  Endif
c788   continue !duumy line for skip=1
c                  print *,'ncoef1=',ncoef1
c                  stop

c                  If(getforce .eq. 1)then !skip, ja is already=ja2
c                  goto 112
c                  Else

c                  print *,'ncoef1=',ncoef1
c                  allocate ja2(ncoef1)
c      do 111 i=1,ncoef1
c                  print *,'ja(i)=',ja(i)
c111               ja2(i)=ja(i)

c                  Endif
c                  print *,'ja2 just after setting ja=',(ja2(ic),ic=1,10)
c                  print *,'ja2(1) =',ja2(1)
c                  pause
c                  print *,''''''''''''''''''''''''''''''''''
c                  pause
c112   Continue !dummy

c                  print *,'ja2 after set to ja =',(ja2(i),i=1,ncoef1)
c                  pause
c                  stop
      deallocate (itempo1)
c
c......sparse "numerical" assembly
c                  print *,'ianal=',ianal
c                  print *,'lumpmass=',lumpmass
      if (ianal .eq. 1) then
cc        allocate ( evalues(neig) )
cc        allocate ( evectors(ndof*neig) )
cc        allocate ( dm(ndof) )
cc        allocate ( elm(maxdofpe,maxdofpe) )
        if (lumpmass .eq. 0) then
cc        allocate ( am(ncoef1) )
        endif
      endif
c
c      print *, 'check point #01'
c      write(6,*) 'check point #01'
c                  print *,'ncoef1 before assembly=',ncoef1
c                  stop

c                  if(it .eq. 2)then
c                  print *,'it =2'
c                  print *,'before assembly2'
c                  stop
c                  endif
32    continue
c32    print *,'dummy line' !dummy line from "if(getforce .eq. 1) above
cvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
c      write(6,*) '*** passed symbass ***'
c                  If(getforce .eq. 1)then
c                  print *,'getforce =1 before assembly'
c                  print *,'ja2(i)=',(ja2(ic),ic=1,10)
c                  endif
c                  pause

                   If(getforce .eq. 1)then
c                  stop
c                  pause
c                  print *,'ncoef1=',ncoef1
c                  write(6,*)'ja(_) before assembly2=',(ja(i),i=1,ncoef1)
c                  print *,'ja(_) before assembly2=',(ja(i),i=1,ncoef1)
```

```
c                     stop
c                     call assembly2(ip,ad,an,lm,ie,je,propmat
c     $,          propsect,elk,be,b,iboundc,ux,uy
c     $,          ae,ia2,ja,x,y,ncoef1,maxdofpe,
c     $ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,tempo1,disp,
c     $iarc,ityp,getforce,it)
c                     print *,'ia2 before assembly=',(ia2(i),i=1,ndof+1)
c                     stop
c                     If(getforce .eq. 1)then
c                     print *,'ja2 before assembly & getforce=1',(ja2(ic),ic=1,10)
c                     endif
c                    pause

c                    print *,'ad stiff before assembly=',(ad(i),i=1,maxdofpe**2)
c                    print *,'ae stiff before assembly=',(ae(i),i=1,maxdofpe**2)
c                    pause
c                    print *,'skip call assembly'
c                    pause
      call assembly(ip,ad,an,lm,ie,je,propmat,
     $          propsect,be,b,iboundc,
     $          ae,ia,ja,x,y,z,ncoef1,maxdofpe,
c     $          ae,ia2,ja,x,y,z,ncoef1,maxdofpe,
     $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c     $ tempo1,disp,iarc,it,area,youngm,esm,qk,itnrmlon)
     $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon,istep,icheck,
     $ zee,dLength,iex)

c                     print *,'ia2 after assembly=',(ia2(i),i=1,ndof+1)
c                     print *,'*** Passed assembly ***'

c                    print *,'ad stiff after assembly=',(ad(i),i=1,maxdofpe**2)
c                    print *,'ae stiff after assembly=',(ae(i),i=1,maxdofpe**2)
c                    pause

c     STOP

                      else
c                    print *,'chk before assembly'
c                    print *,'getstiff=',getstiff
c                    print *,'getforce=',getforce
c                    print *,'Check before assembly'
c                    stop
c                    pause

c                                        call assembly(ip,ad,an,lm,ie,je,propmat
c     $,          propsect,elk,be,b,iboundc,ux,uy
c     $,          ae,ia2,ja,x,y,ncoef1,maxdofpe,
c     $ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,tempo1,disp,
c     $iarc,ityp,getforce,it)

c                     print *,'**ia2 before assembly**=',(ia2(i),i=1,ndof+1)
c                     stop
c                     print *,'going to assembly 2'
c                     print *,'ndofpe=',ndofpe
c                     print *,'chk before assembly'
c                     pause
c                    print *,'B(_) before assembly=',(b(ic),ic=1,ndof)
c                    pause
      call assembly(ip,ad,an,lm,ie,je,propmat,
     $          propsect,be,b,iboundc,
     $          ae,ia,ja,x,y,z,ncoef1,maxdofpe,
c     $          ae,ia2,ja,x,y,z,ncoef1,maxdofpe,
     $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c     $ tempo1,disp,iarc,it,area,youngm,esm,qk,itnrmlon)
     $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon,istep,icheck,
     $ zee,dLength,iex)

c                     print *,'***Passed assembly***'
c                    print *,'ad stiff after assembly=',(ad(ic),ic=1,maxdofpe**2)
c                    print *,'an stiff after assembly=',(an(ic),ic=1,50)
c                    print *,'ae stiff after assembly=',(ae(i),i=1,maxdofpe**2)
c                    pause
c                    stop
c                    write(6,*)'***Passed assembly***'

c                    stop
                     endif


c---------------------------------------------------
c... Only used when solving, getsolve=1
c                    print *,'getforce=?',getforce
c                    print *,'getstiff=?',getstiff
c                    print *,'getsolve=?',getsolve
c                    pause
c                    print *,'B(_)=',(b(ic),ic=1,ndof)
c                    pause
1111                 If(getsolve .eq. 1)Then
                     goto 797
cx     call assembly(ip,ad,an,lm,ie,je,propmat,
cx    $          propsect,be,b,iboundc,
cx    $          ae,ia,ja,x,y,z,ncoef1,maxdofpe,
cx    $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
cx    $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon)
c---------------------------------------------------
c                    print *,'b after assembly getsolve=1 =',(b(i),i=1,ndof)
c                    print *,'ad after assembly getsolve=1',(ad(i),i=1,maxdofpe**2)
c                    print *,'ae after assembly getsolve=1',(ae(i),i=1,maxdofpe**2)
c                    stop
c                    pause
                     Endif



c                    write(6,*)'ae=',(ae(i),i=1,maxdofpe*2)
c                    write(6,*)'disp=',(disp(i),i=1,ndof)
c                    write(6,*)'ae=',(ae(i),i=1,20)
c                    write(6,*)'disp=',(disp(i),i=1,10)
c                    stop
c                    print *,'Getforce after assembly2 = ',getforce
c                    stop
                     If(getforce .eq. 1)then
c                    print *,'b after assembly getforce is 1 =',(b(i),i=1,ndof)
c                    print *,'elk after assembly getforce is 1=',(elk(i),i=1,ndof)
c                    print *,'iboundc=',(iboundc(i),i=1,ndof)
c***need to get support dof into "b"
                     Do 449 I=1,ndof
                     If(iboundc(i) .eq. 1)Then
                     b(i)=0.0
                     Endif
449   Continue
c                    b(1)=0.0
c                    b(2)=0.0
c                    b(3)=0.0
c                    b(4)=0.0
c                    b(6)=0.0
c                    b(7)=0.0
c                    b(8)=0.0
c                    b(9)=0.0
c                    print *,'b modify for support dof =',(b(i),i=1,ndof)
c                    pause
                      goto 45 !this goes to the end of subroutine when we want internal forces
c                    stop
                     endif
c**********
c                     print *,'ae = ',(ae(i),i=1,maxdofpe**2)
c                     stop

c     write(6,*) 'check point #02'
c                    print *,'check point #02'
c                    stop
cc     if (ianal .eq. 1) then
cc     write(6,*) 'diag-mass = dm(-) = ',(dm(i),i=1,ndof)
cc     endif
cc     if (ianal .eq. 1  .and.  lumpmass .eq. 0) then
cc     write(6,*) 'offdiag-mass = am(-) = ',(am(i),i=1,ncoef1)
cc     endif
c
c.....assuming to skip the "REORDERING" process (minimize "fills-in" terms)
c
c                    print *,'***before minfills***'
c                    print *,'iperm before minfills=',(iperm(i),i=1,ndof)
c                    pausc
c                    stop

c                    print *,'iskip inside syseqn=',iskip
c                    pause

                     If(iskip .eq. 1)Then !only do symbass one time, skip after 1st time
c                    print *,'iskip=1 goto 797'
c                    pause
      goto 797
                     Else
     call minfills(ndof,iperm)
c                    print *,'iperm after minfills=',(iperm(i),i=1,ndof)
c                    pause
c_____________________________________________________
                     iskip = 1!This causes symbass to be skipped, only need to go once
c          save for other iterations!!!
c_____________________________________________________
c                    print *,'passed minfills'
c                    print *,'iperm after minfills=',(iperm(i),i=1,ndof)
                     Endif
797   Continue !dummy statement
c                    stop
c.....eigen-solution
cc     if (ianal .eq. 1) then
cc     lump=lumpmass
cc     ishift=0
cc     iprint=1
cc     mtot=20000000
cc     allocate( tempo3(mtot) )
c
c     r(mtot)=working array (in real*8) = tempo3(-)
c     iflag(1,2,3,4,5,6,7,8,9,10)=nreord,loop_unroll,??,??,neig
c                =iprint,lump,ishift,ncoef2,??
c     nreord=0,1,2,3=no reorder,metis,nd,mmd
c
cc     iflag(1)=0          ! 1,2,3 = metis,nd,mmd
cc     iflag(2)=1          ! level of unroll
cc     iflag(5)=neig
cc     iflag(6)=iprint
cc     iflag(7)=lump
cc     iflag(8)=ishift
c     iflag(9)=ncoef2     ! as an output
c
cc     write(6,*) 'inside *cmei*.f: ndof,ncoef1,neig,lump = '
cc     write(6,*)                ndof,ncoef1,neig,lump
cc     write(6,*) 'inside *cmei*.f: ishift,mtot = '
cc     write(6,*)                ishift,mtot
cc     write(6,*) 'inside *cmei*.f: iflag(10) = ',(iflag(i),i=1,10)
cc     write(6,*) 'inside *cmei*.f: ia(-) = ',(ia(i),i=1,ndof+1)
cc     write(6,*) 'inside *cmei*.f: ja(-) = ',(ja(i),i=1,ncoef1)
cc     write(6,*) 'inside *cmei*.f: ad(-) = ',(ad(i),i=1,ndof)
cc     write(6,*) 'inside *cmei*.f: an(-) = ',(an(i),i=1,ncoef1)
cc     write(6,*) 'inside *cmei*.f: dm(-) = ',(dm(i),i=1,ndof)
c
ccc     if (lump .eq. 1) then
ccc     call eigsolver011(ndof,ncoef1,neig,lump,ishift,
ccc    $mtot,ia,ja,ad,an,dm,   evalues,evectors,tempo3,iflag,iperm)
ccc     elseif (lump .eq. 0) then
ccc     call eigsolver022(ndof,ncoef1,neig,lump,ishift,
ccc    $mtot,ia,ja,ad,an,dm,am,evalues,evectors,tempo3,iflag,iperm)
ccc     endif
cc     deallocate( tempo3 )
cc     STOP
cc     endif

                     If(getstiff.eq.1)Then
                     goto 674
                     Endif

c.....sparse "symbolic" factorization
      ncoef2_guess=1000
```

```
c                  print *,'^^^^^^^Chk before allocate itempo1^^^^^^^'
c                  pause
      allocate ( itempo1(ncoef2_guess) )
c                  print *,'ia before symfactd=',(ia(ic),ic=1,12)
c                  print *,'ja before symfactd=',(ja(ic),ic=1,3)
c                  print *,'++++ Chk before symfactd ++++'
c                  print *,'getstiff=',getstiff
c                  print *,'getforce=',getforce
c                  print *,'getsolve=',getsolve
c                  pause
c                  print *,'ndof before symfactd=',ndof
c                  pause
      call symfactd(ndof,ia,ja,iu,itempo1,ip,ncoef2)
c     call symfactd(ndof,ia2,ja,iu,itempo1,ip,ncoef2)

c                  print *,'***************passed symfactd***********'
c     print *, 'after symfactd: IU(-) = ',(iu(ic),ic=1,ndof+1)
c                  pause
c                  pause

c                  write(6,*)'passed symfactd'
c                  stop
c     write(6,*) 'check point #03'
c
c                   print *,'ncoef2=',ncoef2
c                  print *,'NNNNNNNNNNNN ncoef2 NNNNNNNNNNNNN'
c                  pause
c                  stop
      allocate ( ju(ncoef2) )
      allocate ( jut(ncoef2) )
      allocate ( un(ncoef2) )
c                  print *,'itempo1 before copy_int=',(itempo1(ic),ic=1,12)
c                  pause
      call copy_int(ncoef2,itempo1,ju)
c                  print *,'passed copy_int'
c                  print *,'getsolve=1',getsolve
c                  print *,'itempo1 after copy_int=',(itempo1(ic),ic=1,12)
c                  pause
c                  pause

c     print *,'***Passed copy_int***'
c                  print *,'ad stiff after copy_int=',(ad(i),i=1,maxdofpe**2)
c                  print *,'an off-stiff after copy_int=',(an(ic),ic=1,3)
c                  print *,'ae stiff after copy_int=',(ae(i),i=1,maxdofpe**2)
c                  pause

                   If(getsolve.eq.1)then !skip, deallocate
                    allocate (tempo1(10000) )
      deallocate (itempo1)
                   goto 781
                   Endif
      deallocate (itempo1)
781   continue

c
c.....transpose twice to put column numbers in order
      call transad(ndof,ndof,iu,ju,iut,jut)
c                  print *,'*** passed transa  ***'
c     write(6,*) '*** passed transa ***'
c                  print *,'iu after transad=',(iu(i),i=1,ndof+1)
c                  write(6,*)'iu after transad=',(iu(i),i=1,ndof+1)
c                  pause
c                  stop
c
c.....assuming to skip the "SUPER NODE" process (for unrolling purpose)
c
      call supernode(ndof,isupn)
c                  print *,'*** passed supernode  ***'
c                  stop
c
c.....sparse "numerical" factorization
c                  print *,'ia2 before numfa1d= ',(ia2(i),i=1,ndof+1)
c                  write(6,*)'ia2 before numfa1d= ',(ia2(i),i=1,ndof+1)
c                  stop
c                   print *,'ndof before numfa1d=',ndof
c                   pause

c                  print *,'getstiff before numfa1d=',getstiff
c                  print *,'chk before numfa1c'
c                  pause
674                If(getstiff .eq. 1)Then !skip, only need stiff
                   Goto 793
                   Endif

      call numfa1d(ndof,ia,ja,ad,an,iu,
     $           ju,di,un,ip,iup,
     $           isupn,iopf)
c routine numfa1d(n,ia,ja,ad,an,iu,ju,di,un,ip,iup,isupd,iopf)
c     call numfa1d(ndof,ia2,ja,ad,an,iu,
c    $           ju,di,un,ip,iup,
c    $           isupn,iopf)

793   Continue !dummy line
c                  print *,'getstiff after numfa1d=',getstiff
c                  pause

                   If(getstiff .eq. 1)Then !skip, only need stiff
                   Goto 675
                   Endif


c                   print *,'*** PASSED NUMFA1d  ***'
c                  pause
c                  STOP
c     write(6,*) '*** passed numfa1d ***'
c.....sparse forward/backward solution phase

c                   write(6,*)'ndof before fbed=',ndof
c                   write(6,*)'b before fbed=',(b(i),i=1,ndof)
c                   write(6,*)'di before fbed=',(di(i),i=1,ndof)
c                   write(6,*)'iu before fbed=',(iu(i),i=1,ndof+1)
c                   write(6,*)'ju before fbed=',(ju(i),i=1,ncoef2)
c                   write(6,*)'un before fbed=',(un(i),i=1,ncoef2)
c                   print *,'ndof before fbed=',ndof
c                   print *,'b before fbed=',(b(i),i=1,ndof)
c                   print *,'di before fbed=',(di(i),i=1,ndof)
c                   print *,'iu before fbed=',(iu(i),i=1,ndof+1)
c                   print *,'ncoef2=',ncoef2
c                   print *,'ju before fbed=',(ju(ic),ic=1,ncoef2)
c                  pause
c                   print *,'un before fbed=',(un(i),i=1,ncoef2)
c                  stop

c     print *,'////////////////////////////////'
c                  print *,'disp @ before fbed =',(disp(i),i=1,neq)
c                  pausc
c                  print *,'////////////////////////////////'
c                  pause
c                   print *,'going to fbed'
c                  pause
      call fbed(ndof,iu,ju,di,un,b,
     $          tempo1,ioph,isupn,getsolve)

c...tempo1 are displacements!
c                  print *,'tempo1=',(tempo1(i),i=1,neq)
c                   print *,'*** passed fbed  ***'
c                  pause
c                  write(6,*)'*** passed fbed ***'
c*****************************
c                   deallocate (ja)
c                   deallocate (ju)
c*****************************
c                  stop

                   do 40 i=1,ndof
c                  qk(i)=tempo1(i)
40                 disp(i)=tempo1(i)
c                   print *,'tempo1, disp = ',tempo1
c                  print *,'b = ',(b(i),i=1,ndof)
c                  print *,'b after assembly getforce is 1 =',(b(i),i=1,20)
c***need to get support dof into "b"
c                  b(8)=0.0
c                  b(9)=0.0
c                  b(10)=0.0
c                  write(6,*)'b end of syseqn = ',(b(i),i=1,ndof)
c                  print *,'b end of syseqn = ',(b(i),i=1,20)
c                  stop
c     print *,'////////////////////////////////'
c                  print *,'disp @ end of syseqn =',(disp(i),i=1,12)
c                  pause
c                  print *,'////////////////////////////////'
c                  pause
c                  pause
c                  pause
c                  pause
c                  pause
c                  stop
                   If(getforce .eq. 1)then
c                  print *,'disp end syseqn&getforce is 1 =',(disp(i),i=1,ndof)
c                  stop
                   endif
675   continue !blank line used when getstiff=1
45                 continue !blank line used to go if only getting internal forces
c793  continue! dummy,used to go if only get stiffness
                   return
                   end
C*********************************************************************
C***                SUBROUTINE: RESID                              ***
C*********************************************************************
      subroutine resid(dqtot,dqi,dqim1,qk,b,exfor)
      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $           dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $           resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $           dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $           resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c       common/deltaqi/dqi(500)

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
      common/testing/toldis,tolfor,exlk,iconv

      dimension dqtot(*),dqi(*),dqim1(*),b(*),exfor(*),qk(*)

c     dimension exfor(neq),fintk(neq),resk(neq)
c
ccc     print *,'a inside resid=',a
c     print *,'exlk inside resid=',exlk
c     print *,'exfor inside resid=',(exfor(i),i=1,neq)
c     print *,'b inside resid=',(b(i),i=1,neq)

c     print *,'neq inside resid=',neq
c     stop
      do 10 i=1,neq
ccc         resk(i) = exlk*exfor(i) - a(neq+i) !!!For 2D truss only
        resk(i) = exlk*exfor(i) - b(i)
c       print *,'i=',i,'exlk inside resid =',exlk
c       resk(i) = exlk*exfor(i) - fintk(i)
 10   continue
ccc     print *,'exlk inside resid = ',exlk
ccc     print *,'exfor(i) inside resid =',exfor
c     print *,'resk(i) inside resid = ',resk
cc     print *,'exlk inside resid = ',exlk
cc     print *,'exfor(i) inside resid =',(exfor(i),i=1,neq)
cc     print *,'resk(i) inside resid = ',(resk(i),i=1,neq)
cc                  pause

c       stop

c       stop
c
```

```
      return
      end
c ***********************************************************************
c ***              SUBROUTINE: TEST                                   ***
c ***********************************************************************
c     subroutine test(toldis,tolfor,exlk,iconv)
      subroutine test(dqtot,dqi,dqim1,qk,exfor,b,itnrmlon)
      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c         common/deltaqi/dqi(500)

      common/testing/toldis,tolfor,exlk,iconv

      dimension dqtot(*),dqi(*),dqim1(*),exfor(*),qk(*),b(*)

c     dimension dqk(neq),qk(neq),exfor(neq),resk(neq),scali(neq)
c                     print *,'itnrmlon in test=',itnrmlon
cc     print *,'exlk inside TEST=',exlk
c     print *,'tolfor inside TEST=',tolfor
c       stop
      iconv = 0
c
      if (tolfor .lt. 1.d0) then
c     print *,'tolfor = ',tolfor
c     print *,'goto unbalf'
cc     print *,'tolfor, exlk, iconv inside test = ',tolfor,exlk,iconv
c  c     pause
c       stop
c       call unbalf(tolfor,exlk,iconv)
c     print *,'exlk before unbalf = ',exlk
c                     stop
c  c     pause
c     print *,'chk before unbalf in test'
c                     pause
       call unbalf(dqtot,dqi,dqim1,qk,exfor,b)
      end if
c     print *,'toldis after unbalf = ',toldis
c                     pause
c     stop
c
      if (toldis .lt. 1.d0) then
c     print *,'toldis = ',toldis
c     print *,'goto displ'
c     STOP
c       call displ(toldis,iconv)
       call displ(dqtot,dqi,dqim1,qk,h,itnrmlon)
      end if
c
      return
      end
c ***********************************************************************
c ***              SUBROUTINE: UNBALF                                 ***
c ***********************************************************************
c     subroutine unbalf(tolfor,exlk,iconv)
      subroutine unbalf(dqtot,dqi,dqim1,qk,exfor,b)
      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c         common/deltaqi/dqi(500)

      common/testing/toldis,tolfor,exlk,iconv

      dimension dqtot(*),dqi(*),dqim1(*),exfor(*),qk(*),b(*)

c     dimension exfor(neq),resk(neq),scali(neq)
c
c     print *,'Start of "UNBALF"'
c     print *,'exlk inside UNBALF=',exlk
c       stop
      unbfi = 0.d0
      unbfe = 0.d0
c
c ... compute numerator of eqn. 4.133
c ... compute denominator of eqn. 4.133
c     print *,'scali(i) inside UNBALF"=',(scali(i),i=1,neq)
c     print *,'resk(i) inside UNBALF"=',(resk(i),i=1,neq)
c                     pause
c     print *,'exfor(i) inside UNBALF"=',(exfor(i),i=1,neq)
c       stop
      do 10 i=1,neq
        unbfi = unbfi + dabs(scali(i))*resk(i)**2
        unbfe = unbfe + dabs(scali(i))*exfor(i)**2
 10   continue
c
c ... compute eqn. 4.133
c     print *,'unbfi=',unbfi
c     print *,'unbfe=',unbfe
c                     print *,'scali(_)=',(scali(ic),ic=1,neq)
c     print *,'exlk=',exlk
      delf = dsqrt(unbfi)/dsqrt(unbfe)/exlk
c     print *,'delf inside unbal = ',delf
c                     pause
C     stop
c

c ... test to see if delf is greater than tolfor
      if (dabs(delf) .gt. tolfor) then
        iconv = iconv + 100
      end if
c     print *,'iconv inside unbal = ',iconv
c     stop
c
      return
      end
c ***********************************************************************
c ***              SUBROUTINE: DISPL                                  ***
c ***********************************************************************
c     subroutine displ(toldis,iconv)
      subroutine displ(dqtot,dqi,dqim1,qk,b,itnrmlon)
      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c         common/deltaqi/dqi(500)

      common/testing/toldis,tolfor,exlk,iconv
      commnon/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2

                     dimension dqtot(*),dqi(*),dqim1(*),qk(*),b(*)

c     dimension dqk(neq),qk(neq)
c
c     print *,'neq inside displ=',neq
c       stop
      dqsq = 0.d0
      qsq = 0.d0
c
c ... compute deld using eqn. 4.132
      do 10 i=1,neq

                     If(itnrmlon .eq. 0)then
                     print *,'dqi(i)=',dqi(i)
        dqsq = dqsq + dqi(i)**2
                     Else !iteration on normal
                     print *,'dqk(i)=',dqk(i)
        dqsq = dqsq + dqk(i)**2
                     End if
c       qsq = qsq + qk(i)**2
        qsq = qsq + b(i)**2
c       qsq = qsq + a(i)**2
 10   continue
c     print *,'dqsq inside displ =',dqsq
c     print *,'qsq inside displ =',qsq
c       stop
      deld = dsqrt(dqsq)/dsqrt(qsq)
c     print *,'deld inside displ = ',deld
c                     pause
c     stop
c
c ... test to see if deld is greater than toldis
      if (deld .gt. toldis) then
        iconv = iconv + 10
      end if
c     print *,'iconv inside displ = ',iconv
c     stop
c
      return
      end
C ***********************************************************************
C ***              SUBROUTINE: RESULT                                 ***
C ***********************************************************************
      subroutine result(istep,it,dqtot,dqi,dqim1,qk,exfor,ad)
      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c         common/deltaqi/dqi(500)

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
      common/testing/toldis,tolfor,exlk,iconv

                     dimension dqtot(*),dqi(*),dqim1(*),exfor(*),qk(*),ad(*)

ccc     common/disp/ux(2),uy(2),disp(500)

cccc     print *,'ux(1) & ux(2) =',ux(1),ux(2)
cc     print *,'In Result'
cc                     pause
c     stop
c     dimension exfor(neq),qk(neq)
c
c     if (istep .eq. 1) then
c  c       write(*,10)
c  c       write(*,20)
c       write(11,10)
c       write(11,20)
c     end if
c
      open(15,file='fea3d2_out.txt',status='unknown')
c     open(12,file='fea3d2_out_F.txt',status='unknown')
c     open(13,file='fea3d2_out_q1.txt',status='unknown')
c     open(14,file='fea3d2_out_q2.txt',status='unknown')
c     write(*,50) '*',istep,exlk,qk(1),qk(2),it
```

```
c                    print *,'********** Inside Result **************'

c     Do 51 i=1,neq
c     xinternalforce = exlk * exfor(3)
c     yinternalforce = exlk * exfor(4)
caa     appliedforce = exlk * exfor(5)
carch     appliedforce = exlk * exfor(62) !arch
c                    print *,'exlk=',exlk
c                    print *,'exfor(62)=',exfor(62)
c                    pause
c     appliedforce = exlk * exfor(6)

c     aforce=a(12)
c     appliedforce = exlk * exfor(15)
c     appliedforce = exlk * exfor(3)
c18     appliedforce = exlk * exfor(20)
carch     appliedforce = exlk * exfor(44)
cx     print *,'exlk inside result = ',exlk
cccc     print *,'exfor(12) inside result = ',exfor(12)
cc     print *,'exfor(15) inside result = ',exfor(15)
c     print *,'exfor(3) inside result = ',exfor(3)
c     print *,'appliedforce inside result=',appliedforce
cc     print *,'a(4) displacement = ',a(4)
cc     print *,'a(3) displacement = ',a(3)
c     pause
c     stop
cccc     write(*,50) '*',istep,exlk,xinternalforce,yinternalforce,
cccc     c              a(3),a(4),it
ccc     write(*,50) '*',istep,exlk,xinternalforce,yinternalforce,
ccc     c              a(3),a(4),it
c     write(*,50) '*',istep,exlk,appliedforce,
c     c              a(3),a(4),it

c     write(12,55) appliedforce
c 55     format(f12.4)
c     write(14,56) a(6)
c 56     format(f12.4)
c     write(15,50) '*',istep,exlk,appliedforce,a(3),it
c 50   format(a1,1x,i3,1x,f10.6,1x,f10.6,1x,f10.6,1x,f10.6,1x,f10.6,
c     c          1x,i3)
c 50   format(a1,1x,i3,1x,f12.4,1x,f12.4,1x,f12.4,1x,f12.4,1x,
c     c      f12.4,1x,f12.4,1x,i3)
c below for T3
c 50   format(a1,1x,i3,1x,f12.4,1x,f12.4,1x,f12.4,1x,f12.4,1x,
c     c      f12.4,1x,f12.4,1x,i3,1x,f12.4)
c below for T3x
ccc 50   format(a1,1x,i5,1x,f12.4,1x,f16.0,1x,f12.4,1x,f12.4,1x,
ccc     c      f12.4,1x,f12.4,1x,f12.4,1x,f12.4,2x,i5)
c 50   format(a1,1x,i3,1x,f10.6,1x,f12.4,1x,f10.6,1x,f10.6,1x,
c     c      f10.6,1x,f10.6,1x,f10.6,1x,f10.6,1x,i3)
c 50   format(a1,1x,i3,1x,f10.6,1x,f10.6,1x,f10.6,1x,f10.6,
c     c          1x,i3)
c     print *,'check inside result'

c     if(istep .eq. 1) then
c     write(12,55)
c 55   format('Appliedforce')
c     write(13,56)
c 56   format('Two ele Truss')

carch     write(15,53)
carch 53   format(' istep     exlk       appliedforce     dqtot(61)
carch     $ dqtot(62)      it')

c 53   format(' istep     exlk       appliedforce     dqtot(4)
c     $ dqtot(5)      it')

c 53   format(' istep     exlk     appliedforce     a(3)     a(4)
c     $a(5)     a(6)   it')
c 53   format(' istep     exlk       appliedforce       a(1)
c     $     a(2)       a(3)     a(4)     a(5)     a(6)     it')

c     write(*,53)
c 53   format(' istep     exlk     appliedforce     a(3)     a(4)
c     $a(5)     a(6)     a(9)     a(10)     it')
c 53   format('  istep     exlk     appliedforce     a(5)       a(6)
c     $       a(13)       a(14)       a(11)       a(12)   it')
c18 53   format('  istep     exlk     appliedforce     a(15)
c18     $ a(16)       a(17)       a(18)       a(19)       a(20)  it')
CCC 53   format('  istep     exlk     appliedforce       a(19)
CCC     $ a(20)       a(41)       a(42)       a(43)       a(44)   it')
carch 53   format('  istep     exlk     appliedforce       a(47)
carch     $ a(48)       a(39)       a(40)       a(43)       a(44)   it')
ccc 53   format('  istep     exlk     appliedforce       a(10)
ccc     $ a(11)       a(12)     it')
c 53   format('  istep     exlk     appliedforce       a(13)
c     $ a(14)       a(15)     it')
cc 53   format('  istep     exlk     appliedforce       a(1)
cc     $     a(2)       a(3)       it')
cdome 53   format('  istep     exlk     appliedforce       a(6)
cdome     $ a(9)       a(3)     it')
c 53   format(' istep     exlk     appliedforce     a(3)     a(4)
c     $a(5)     a(6)   it')
cc below for T3
cc 53   format(' istep     exlk     appliedforce     a(3)       a(4)
cc     $ a(5)       a(6)       it       a(12)')
c below for T3x
cxxx 53   format(' istep     exlk     appliedforce     a(3)       a(4)
cxxx     $ a(5)       a(6)       a(9)       a(10)       it')
c below for T3y
c 53   format(' istep     exlk     appliedforce     a(1)       a(2)
c     $ a(3)       a(4)       it       a(12)')
c 53   format(' istep     exlk     xinfor     yinfor     a(3)     a(4)
c     $     it')
c     endif

c     print *,'check 2 inside result'
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(3),a(4),a(5),a(6),a(11),a(12),it
c Below IS FOR T3
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(3),a(4),a(5),a(6),it,a(12)
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(4),a(5),a(6),it
c6     write(15,50) '*',istep,exlk,appliedforce,
c6     c              a(1),a(2),a(3),a(4),a(5),a(6),it

c Below IS FOR T3x
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(5),a(6),a(13),a(14),a(11),a(12),it
c18     write(15,50) '*',istep,exlk,appliedforce,
c18     c              a(15),a(16),a(17),a(18),a(19),a(20),it
c18 50   format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,f10.4,
c18     c          f10.4,6x, f10.4, f10.4,6x,i3)
CCC     write(15,50) '*',istep,exlk,appliedforce,
CCC     c              a(19),a(20),a(41),a(42),a(43),a(44),it
carch     write(15,50) '*',istep,exlk,appliedforce,
carch     c              a(39),a(40),a(47),a(48),a(43),a(44),it
carch 50   format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,3X,f10.4,
carch     c          f10.4,3x, f10.4,3X f10.4,4x,i5)
cc     print *,istep,exlk,appliedforce,
cc     c              a(10),a(11),a(12),it
ccc     write(15,50) '*',istep,exlk,appliedforce,
ccc     c              a(10),a(11),a(12),it

c 50   format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,f10.4,
c     c          6x,i3)
c6 50   format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,4x,f10.4,
c6     c          f10.4,f10.4,f10.4,4x,i3)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(13),a(14),a(15),it
cc     write(15,50) '*',istep,exlk,appliedforce,
cc     c              a(1),a(2),a(3),it
caa     write(15,50) '*',istep,exlk,appliedforce,
caa     c              dqtot(4),dqtot(5),it

c *** frame ***
c** "frame_in.dat" & "frame_out.dat**************************************************

cf                    totald=sqrt(dqtot(1)**2+dqtot(2)**2) !total nodal displ.

cframe                appliedforce = exlk * exfor(1)

cframe     if(istep .eq. 1) then
cframe     write(15,53)
c  53  format(' istep     exlk     appliedforce       dqtot(1)
c     $   dqtot(2)     dqtot(4)  dqtot(5)   it')
cframe  53  format(' istep     exlk     appliedforce       dqtot(1)
cframe     $   dqtot(2)     dqtot(4)     dqtot(5)       it')

cframe     endif


c     write(15,50) '*',dqtot(1)

cframe     write(15,50) '*',istep,exlk,appliedforce,
cframe     c dqtot(1),dqtot(2),dqtot(4),dqtot(5),it
cframe 50   format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,6x,
cframe     c          e10.4,6x,e10.4,3X,i5)

c *** Arch ***
c** "arch42in.dat" & "arch42_3D_in.dat**************************************************
carch     appliedforce = exlk * exfor(65)
carch     if(istep .eq. 1) then
carch     write(15,53)
c  53  format(' istep     exlk       appliedforce     dqtot(11)
c     $     it')
c     endif
c     write(15,50) '*',istep,exlk,appliedforce,
c     c dqtot(11),it
c  50   format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,4x,i5)

ccc     write(15,50) exlk
ccc  50   format(e14.6)

carch  53  format(' istep     exlk     appliedforce       dqtot(61)
carch     $   dqtot(62)   dqtot(64)  dqtot(65)  dqtot(85)   dqtot(86)
carch     $     it  ad(61)   ad(62)')
carch     endif
carch     write(15,50) '*',istep,exlk,appliedforce,
carch     c dqtot(61),dqtot(62),dqtot(64),dqtot(65),dqtot(85),dqtot(86),it,
carch     c  ad(62),ad(65)
carch  50   format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
carch     c          3x,e10.4,3x, e10.4,3X e10.4,4x,i5,4x,e10.4,3x,e10.4)

c** "shallow_truss_initial.dat"**************************************************
cshall     appliedforce = exlk * exfor(38)
cshall     appliedforce2 = exlk * exfor(20)
c                    print *,'exfor(20)',exfor(20)
c                    print *,'exfor(38)',exfor(38)
c                    pause
cc     weightotal=0.d0
cc                   Do 55 i=1,nel
c                    print *,'istep=',istep
c                    print *,'i=',i
c                    print *,'weightotal=',weightotal
c                    print *,'weight(_)=',weight(i)
c                    pause
cd     SEDratio(i)=strainD(i)/weight(i)
cc                   weightotal=weightotal+weight(i)
cc55                 Continue

cshall     if(istep .eq. 1) then
cshall     write(15,53)
cshall 53   format(' istep     exlk     appliedforce38   appliedforce20
cshall     $   a(20)     a(38)     it')
cshall               endif

cshall     write(15,50) '*',istep,exlk,appliedforce,appliedforce2,
cshall     c   dqtot(20),dqtot(38),it

cshall 50   format(a1,2x,i4,2x,e14.6,4x,e12.6,4x,e12.6,4x,e12.6,4x,e12.6,
```

```
cshall    c 1x,i5)

c     write(20,60) dqtot(20)
c60   format(e14.6)
cc                  If(iprint .eq. 0)then !do this first time only
c     write(19,61) exlk,dqtot(20),dqtot(38)
c61   format(e14.6,2x,e14.6,2x,e14.6)
cc                  iprint=1
cc                  iprt=1
cc                  endif

c                   If(iprint/10 .eq. 10*iprt/100)then
c...output every 10th data point
cc                  If(iprint .eq. iprt)then
cc     write(19,62) exlk,dqtot(20),dqtot(38)
cc62   format(e14.6,2x,e14.6,2x,e14.6)
c                   print *,'iprint=',iprint
c                   print *,'iprint/10=',iprint/10
c                   print *,'iprt=',iprt
c                   print *,'10*iprt/100=',10*iprt/100
c                   pause
cc                                  If(iprint .gt. 1)then
cc                                   iprt=iprint+10
cc                                  Else
cc                        iprt=iprint+9
cc                                  Endif
cc                  Endif
c                   print *,'iprint 2=',iprint
c                   print *,'iprt 2=',iprt
c                   pause
cc                  iprint=iprint+1


c*********************************************************************
c** "t3a_in.dat" 3 member truss ****************************
c     appliedforce = exlk * cxfor(4)
c     if(istep .eq. 1) then
c     write(15,53)
c 53  format(' istep     exlk     appliedforce          dqtot(4)
c    $   dqtot(5)    dqtot(6)  dqtot(7)    dqtot(8)    dqtot(9)
c    $      it')
c     endif
c     write(15,50) '*',istep,exlk,appliedforce,
c    c dqtot(4),dqtot(5),dqtot(6),dqtot(7),dqtot(8),dqtot(9),it
c50   format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,c10.4,3X,c10.4,
c     c        3x,e10.4,3x, e10.4,3X e10.4,4x,i5)
c*********************************************************************

c     if(istep .eq. 1) then
c     write(15,53) '*',istep,exlk,appliedforce,
c     c              dqtot(61),dqtot(62),it
c 53  format(a1,2x,i3,5x,f10.6,3x,f10.6,6x,f10.8,6x,f12.8,
c     c         6x,i3)
c                   endif
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              dqtot(61),dqtot(62),it
c 50  format(a1,2x,i3,5x,f10.6,3x,f10.6,6x,f10.8,6x,f12.8,
c     c         6x,i3)
c** "Tripodin.dat  using 3d element"*******************************
c       appliedforce1 = exlk * cxfor(10)
c       appliedforce2 = exlk * cxfor(12)
c     if(istep .eq. 1) then
c     write(15,53)
c 53  format(' istep     exlk       appforcex   appforcez
c     c    a(10)          a(11)          a(12)          it')
c     endif
c     write(15,50) '*',istep,exlk,appliedforce1,appliedforce2,
c     c              dqtot(10),dqtot(11),dqtot(12),it
c50   format(a1,2x,i4,5x,f10.6,3x,c10.3,3x,e10.3,6x,e10.4,6x,e10.4,6x,
c     c      c10.4,4x,i5)
c** "twobar_in.dat"*************************************************
ctwo        appliedforce = exlk * cxfor(7)
ctwo      if(istep .eq. 1) then
ctwo      write(15,53)
ctwo 53   format(' istep     exlk       appliedforce       a(5)
ctwo     $    a(7)       it          ad(5)       ad(7)')
ctwo      endif
ctwo      write(15,50) '*',istep,exlk,appliedforce,
ctwo      c     dqtot(5),dqtot(7),it,
ctwo      c     ad(5),ad(7)
ctwo50    format(a1,2x,i4,5x,e14.6,3x,e10.4,6x,e10.4,4x,c10.4
ctwo      c          ,4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
c** "snap4_in.dat"*************************************************
csnap4       appliedforce = exlk * cxfor(11)
csnap4      if(istep .eq. 1) then
csnap4      write(15,53)
csnap4 53   format(' istep     exlk       appliedforce       a(1)
csnap4     $    a(2)       a(11)       it          ad(3)       ad(4)')
csnap4      endif
csnap4      write(15,50) '*',istep,exlk,appliedforce,
csnap4      c     dqtot(1),dqtot(2),dqtot(11),it,
csnap4      c     ad(1),ad(2)
csnap450    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
csnap4      c          4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
c** "snap_in.dat"*************************************************
csnap        appliedforce = exlk * cxfor(11)
csnap      if(istep .eq. 1) then
csnap      write(15,53)
csnap 53   format(' istep     exlk       appliedforce       a(1)
csnap     $    a(2)       a(11)       it          ad(3)       ad(4)')
csnap      endif
csnap      write(15,50) '*',istep,exlk,appliedforce,
csnap      c     dqtot(1),dqtot(2),dqtot(11),it,
csnap      c     ad(1),ad(2)
csnap50    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
csnap      c          4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
c** "domein.dat"*************************************************
cdome        appliedforce = exlk * cxfor(3)
cdome      if(istep .eq. 1) then
cdome      write(15,53)
cdome 53   format(' istep     exlk       appliedforce       a(1)
cdome     $    a(2)       a(3)        a(4)        a(5)       a(6)
cdome     $    it         ad(3)       ad(4)')
cdome      endif
cdome      write(15,50) '*',istep,exlk,appliedforce,
cdome      c     dqtot(1),dqtot(2),dqtot(3),dqtot(4),dqtot(5),dqtot(6),it,
cdome      c     ad(3),ad(4)
cdome50    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
cdome      c          3x,e10.4,3x, e10.4,3X e10.4,4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
c** "truss7_3d_in.dat  using 3d element"*******************************
C** "truss11_3d_in.dat using 3d element"*******************************
ctruss       appliedforce = exlk * cxfor(8)
ctruss      if(istep .eq. 1) then
ctruss      write(15,53)
ctruss 53   format(' istep     exlk       appliedforce       a(1)
ctruss     $    a(2)       a(4)        a(5)       a(7)       a(8)
ctruss     $    it')
ctruss      endif
ctruss      write(15,50) '*',istep,exlk,appliedforce,
ctruss      c          dqtot(1),dqtot(2),dqtot(4),dqtot(5),dqtot(7),dqtot(8),it
ctruss50    format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,3X,f10.4,
ctruss      c          f10.4,3x, f10.4,3X f10.4,4x,i5)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(6),a(9),a(3),it
c 50  format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,
c     c         6x,i3)
cc 50  format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,f10.4,
cc    c         6x,i3)
c     print *,'check 3 inside result'
c     write(*,50) '*',istep,exlk,appliedforce,
c     c              a(3),a(4),a(5),a(6),a(13),a(14),it
c Below IS FOR T3y
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(1),a(2),a(3),a(4),it,a(12)
c     write(15,50) '*',istep,exlk,xinternalforce,yinternalforce,
c     c              a(3),a(4),it

c      stop
ccc    write(12,51) xinternalforce,yinternalforce
ccc 51  format(f12.2,1x,f12.2)
c     write(12,51) appliedforce
c 51  format(f12.2)

c     write(13,52) a(3)
cc    write(13,52) a(5)
c18   write(13,52) a(20)
carch     write(13,52) a(44)
cc 52  format(f10.6)


c 51  continue
c
cc 10  format(t5,'Load',t13,'Load',
cc    $       t28,'External',t62,'No. of')
cc 20  format(t5,'Step',t13,'Multiplier',
cc    $       t28,'Load',t45,'Displacement',
cc    $       t62,'Iter.')
cc 30  format(t5,i3,t13,f10.6,t28,f10.6,t45,f12.6,t62,i3)
c
c      print *,'end of result'
c      pause
c      stop
c** "func_in.dat"**************************************************
      appliedforce1 = exlk * cxfor(1)
      appliedforce2 = exlk * cxfor(2)
c                   print *,'result'
c                   print *,'cxlk=',cxlk
c                   print *,'cxfor(1)=',cxfor(1)
c                   print *,'cxfor(2)=',cxfor(2)
c                   pausc
    if(istep .eq. 1) then
    write(15,53)
 53  format('   istep      exlk       appliedforce1  appliedforce2
   $      a(1)       a(2)          it       ad(5)       ad(7)')
    endif
    write(15,50) '*',istep,exlk,appliedforce1,appliedforce2,
    c     dqtot(1),dqtot(2),it,
    c     ad(5),ad(7)
50  format(a1,2x,i6,5x,e16.8,3x,e10.4,6x,e10.4,4x,e10.4
    c         ,4x,e10.4,4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
    return
    end

C*********************************************************************
C***              SUBROUTINE: ITNRML                           ***
C*********************************************************************
c     subroutine itnrml(itemax,toldis,tolfor,dli,dlim1,
c    $              exlk,iconv,iflag,it,ne,iplvl)
    subroutine itnrml(itemax,dli,dlim1,b,exfor,ad,ncltype,qk,
   $              iflag,it,ne,iplvl,ndof,ia,nel,ncocf1,jcall,
c    $              iflag,it,nc,iplvl,ndof,ia1,ia2,nel,ncocf1,jcall,
   $              icall,ie,jc,iet,jct,maxdofpe,dqtot,dqi,dqim1,
   $              ja,an,propmat,propsct,x,y,z,ndofpn,numnodes,
   $              ip,iu,disp,ianal,ndofpe,iskip,itcmpo2,iarc,lm,
c    $              itnrmlon,iboundc,esin,elk)
   $              itnrmlon,iboundc,ae,di,istep,iex)

c                    usc

    implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
C     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
C    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
C    $          resk(50),scali(50),work1(50),indx(50),neq
    common/RIK/ stifi(20,20),dqk(5000),dqk1(5000),
   $     dqk2(5000),fintk(5000),
```

```
     $    resk(5000),scali(5000),work1(5000),indx(5000),neq

c         common/dcltaqi/dqi(500)

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
      common/disp/ux(2),uy(2),uz(2)
      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/zvector/za(1500),za1(1500),za2(1500),za3(1500),ipick
      common/testing/toldis,tolfor,exlk,iconv

      dimension ie(*),je(*),iet(*),jet(*),b(*),exfor(*),ncl(*)
      dimension dqtot(*),dqi(*),dqim1(*),ad(*),qk(*)
c     dimension dqtot(*),dqi(*),dqim1(*),qk(*),ia2(*),ad(*)
                dimension ja(*),an(*),disp(*)
c               dimension an(*),disp(*)
                dimension propmat(1),propsect(1),x(*),y(*),ip(*),iu(*)
      dimension itempo2(*),lm(1),ae(1)
                dimension iboundc(*)
c     dimension itempo2(*),lm(1), elk(maxdofpe,maxdofpe)

c     dimension stifi(neq,neq),dqi(neq),dqim1(neq),dqk(neq),
c     $         dqk1(neq),dqk2(neq),exfor(neq),fintk(neq),qk(neq),
c     $         resk(neq),scali(neq),work1(neq)
c
c                 deallocate (ja)

      itnrmlon=1 !Let everyone know we are going thru "itnrml"
c             "syseqn" "& "assembly" need to do something more

c     print *,'!!!!!!!!!!!!!!!!  starting itnrml  !!!!!!!!!!!!!!!!'
c               pause
c               print *,'itempo2(1) inside itnrml=',itempo2(1)
c               pause

c               print *,'ad stiff=',(ad(i),i=1,maxdofpe**2)
c               pause

c     print *,'ipick = ',ipick
c               print *,'istep=',istep
c               print *,'it=',it
c     print *,'getforce =',getforce
c     print *,'getstiff =',getstiff
c                print *,'qk(-) start of itnrml=',(qk(i),i=1,ndof)
c                print *,'b start of itnrml=',(b(i),i=1,ndof)
c                print *,'ad(_) start itnrml=',(ad(ic),ic=1,ndof)
c                print *,'an(_) start itnrml=',(an(ic),ic=1,50)
c               pause
c       stop
c     print *,'a at start of itnrml = ',a
c     print *,'exlk at start of itnrml = ',exlk
c     print *,'dqi at start of itnrml = ',(dqi(i),i=1,ndof)
c      print *,'exfor at start of itnrml =',(exfor(i),i=1,ndof)
c               pause
c       stop
      dlk = 0.d0
      iconv = 1
      iflag = 0
c      it = 0
c
      do 20 i=1,neq
c        do 10 j=1,neq
c          stifi(j,i) = 0.d0
c 10     continue
c        print *,'exfor(i) start =',exfor(i)
         dqk(i) = 0.d0
         dqk1(i) = 0.d0
         dqk2(i) = 0.d0
         fintk(i) = 0.d0
         work1(i) = 0.d0
c        print *,'exfor(i) end =',exfor(i)
 20   continue
c** set counter equal to zero for next "if"
      iam = 0
c     print *,'exfor =',exfor
c     stop
c     print *,'iconv = ',iconv
c     print *,'itemax = ',itemax
c     print *,'it = ',it
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1002 if (iconv .ne. 0 .and. it .le. itemax) then
c        print *,'iconv =',iconv
c        Print *,'it = ',it
c        print *,'itemax =',itemax
c                 stop
         it = it + 1
c        write(*,*) 'Beginning Iteration Number:',it
c                 stop
c ... compute Ktk for eqn. 4.97
cc       print *,'ne & iplvl inside itnrml = ',ne,iplvl
c        print *,'qk inside itnrml=',qk
c c      pause
c        stop
c        Getforce=1
c**Only get stiffness
                  Getsolve=0
                  Getforce=0
         Getstiff=1
c        print *,'exfor = ',exfor
c                 print *,'chk before syseqn inside itnrml'
c                 write(6,*)'chk before syseqn inside itnrml'
c        stop

c                 deallocate (ja)
c                 deallocate (itempo1)
c                 deallocate (an)
c                 print *,'********** iskip in itnrml **************'
c                 print *,'********** iskip in itnrml **************'
c                pause
c                 print *,'iskip in itnrml before 1st syseqn=',iskip
c                pause
c                 print *,'iskip in itnrml before 1st syseqn=',iskip
c                 pause
c                 print *,'iskip in itnrml before 1st syseqn=',iskip
c                 pause
c                 print *,'iskip in itnrml before 1st syseqn=',iskip
c                 pause
c                 print *,'getforce in itnrml before 1st syseqn=',getforce
c                 print *,'getstiff in itnrml before 1st syseqn=',getstiff
c     pause
c                 print *,'********************'
c                 print *,'********************'
c                 print *,'disp before 1st syseqn in itnrml=',(disp(i),i=1,ndof)
c                 print *,'********************'
c                 print *,'********************'
c                 pause
c                 print *,'ad stiff before=',(ad(i),i=1,ndof)
c                 pause
c                 print *,'an stiff=',(an(i),i=1,maxdofpe**2)
c                 print *,'ae(_)before 1st syseqn in itnrml=',(ae(ic),ic=1,18)
c                 print *,'b(_) before=',(b(ic),ic=1,ndof)
c                 pause
c     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                 pause
c                 pause

                  call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $       ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $       neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $       iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $       ux,uy,uz,di,it,area,youngm,
     $       ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $ zce,dLength,iex)

c     $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c       print *,'dqk1 after syseqn in intml=',dqk1
c                 print *,'********************'
c                 print *,'********************'
c                 print *,'disp after 1st syseqn in itnrml=',(disp(i),i=1,ndof)
c                 pause
c                 print *,'b after 1st syseqn in itnrml=',(b(i),i=1,ndof)
c                 pause
c                 print *,'ad stiff after 1st syseqn=',(ad(i),i=1,ndof)
c                 pause
c                 print *,'an(_) after 1st syseqn in itnrml=',(an(ic),ic=1,50)
c                 print *,'ae stiff after 1st syseqn=',(ae(i),i=1,maxdofpe**2)
c                 print *,'********************'
c                 print *,'********************'
c                 pause

c     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                 pause

c        If(it .eq. 2)then
c     print *,'it = ',it
c                 print *,'disp after 1st syseqn in itnrml=',(disp(ic),ic=1,ndof)
c                 print *,'b(_) after 1st syseqn in itnrml=',(b(ic),ic=1,ndof)
c                 print *,'ad stiff after 1st syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                 print *,'an(_) after 1st syseqn in itnrml=',(an(ic),ic=1,50)
c        stop
c        endif

c                 stop
c     pause
c                  print *,'chk after syseqn inside itnrml'
c                  write(6,*)'chk after syseqn inside itnrml'
c                 write(6,*)'b after 1st syseqn in itnrml=',(b(i),i=1,neq)
c                 write(6,*)'disp after 1st syseqn in itnrml=',(disp(i),i=1,neq)

c       stop

c     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                 pause

c^^^^^ SETTING ZERO DIAGONAL VALUES TO ONE ^^^^^^^^^^^^^^^^
c       Do 1014 i=neq*2+1,neq*3
        Do 1014 i=1,neq
c        If(ad(i) .GT. .00001) GoTo 1014
        If(ad(i) .GT. 1.e-8) GoTo 1014
c         ad(i)=1.e-8
c         ad(i)=0.d0
1014    Continue

c                 print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                 pause

c???          do 702 i=1,neq
c              scali(i) = 1.d0/stifi(i,i)
c     c c     print *,'a(neq*2+i) = ',a(neq*2+i)
cc               scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
c???             scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c???              print *,'scali =',scali(i)
c???              write(6,*)'scali =',scali(i)
c??? 702        continue

c**1 call modify to get the stiffness w/delected rows/columns
cc        call modify
c         print *,'dqk1 after modify in intml=',dqk1
c         stop
c     print *,'Check after fifth modify'
c     print *,' "a" after 5th modify = ',a
c     c      pause
c c     print *,'ibstor = ',ibstor
c c     print *,'bvstor = ',bvstor
c       stop

        Getstiff = 0 ! set getting just stiff back to zero

cc      print *,'a after syseqn inside itnrml = ',a
c         If(it .eq. 2)then
c     print *,'it = ',it
c     print *,'a at it=2 ',a
c         stop
```

```
c         endif
c       call stiff

cc      print *,'stifi inside itnrml = ',stifi
c ... compute (dqk)1 using eqn. 4.97
c ... compute (dqk)2 using eqn. 4.99
c       print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                       pause
c        stop

         do 30 i=1,neq
           dqk1(i) = exfor(i)
c          dqk1(i) = b(i)
           dqk2(i) = resk(i)
c        print *,'exfor(i) = ',exfor(i)
c       print *,'dqk1(i) & i = ',dqk1(i),i
c       print *,'resk(i) & i = ',resk(i),i
c       print *,'dqk2(i) & i = ',dqk2(i),i
 30      continue
c                       pause
c       print *,'dqk1(_)=',(dqk1(ic),ic=1,neq)
c                       pause
c       print *,'dqk2(_)=',(dqk2(ic),ic=1,neq)
c       pause
c         stop

c c     print *,'resk = ',resk

c put dqk1(i) into big "a(i) as a force, get out as displacemnet"
        j=0
cc      print *,'neq, neq2, np2 = ',neq,neq2,np2
c        stop
c        print *,'a =',a
c       stop
c**********************************************
c       If(ityp .eq. 6)Then
cc        do 31 i=neq+1,neq*2

         do 31 i=1,neq
cc          j=j+1
c          a(i)=dqk1(j)

           b(i)=dqk1(i)
c                       print *,'b(i) before new 2nd syscqn=',b(i)
c                       pause
c c     print *,'a(i) i = ',a(i),i
c       print *,'dqk1(i) = ',dqk1(i)
 31      continue
c       print *,'b(_) after dqk1(_) = ',(b(ic),ic=1,neq)
c                       pause

CXX                     write(6,*)'b(i) insidc itnml=',(b(i),i=1,neq)
c       Endif
c       If(ityp .eq. 12)Then
c        do 34 i=neq+1,neq*3
c          j=j+1
c          a(i)=dqk1(j)
c c     print *,'a(i) i = ',a(i),i
c c     print *,'dqk1(j) j = ',dqk1(j),j
c 34     continue
c       Endif
c**********************************************

c       print *,'dqk1 = ',dqk1
c       stop
c       call solve(dqk1)
cc  c     print *,'n before ludcmp = ',n
c          stop
cc  c     print *,'****ludcmp 3****'
c       c     pause

cc       write(*,*) 'Before dcmpbd inside itnrml'
c       print *,'a = ',a
c       stop
c** solve for new "dqk1"

        ipick = 1

cc       call dcmpbd
cc       print *,'after dcmpbd'
c        print *,'a after dempbd inside itnrml =',a
c        stop
cc       write(*,*) 'After dcmpbd inside itnrml'
cc       call slvbd
c       write(*,*) 'After slvbd1 & ipick=1 inside itnrml'
c        stop
c                       print *,'############ NEW SYSEQN ADDED itnrml############'
c                       print *,'############ 2nd Syseqn in itnrml ############'
c                       pause

c                       print *,'*******************'
c                       print *,'*******************'
c                       print *,'disp before 2nd syseqn in itnrml=',(disp(i),i=1,ndof)
c                       pause
c                       print *,'b before 2nd syseqn in itnrml=',(b(i),i=1,ndof)
c                       pause
c                       print *,'ad stiff before 2nd syseqn=',(ad(i),i=1,maxdofpe**2)
c                       pause
c                       print *,'an(_) before 2nd syseqn in itnrml=',(an(ic),ic=1,50)
c                       pause
c                       print *,'ac stiff after 1st syseqn=',(ac(i),i=1,maxdofpe**2)
c                       print *,'*******************'
c                       print *,'*******************'
c
c... 1 only want to solve, 1 don't want update forces or stiff
                          getforce=0
                          getstiff=0
                          getsolve=1
c                       print *,'getsolve=1 inside itnrml before 2nd syseqn'
c                       pause
c...Need to solve eqns only, no new loads or stiff!
        call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $            ieall,ie,jc,ict,jet,jeall,ia,ac,maxdofpe,
     $            neltypc,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $            iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $            ux,uy,uz,di,it,area,youngm,
     $         ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $ zee,dLength,iex)
c                       print *,'############## END SYSEQN ##################'
c                       pause
c                       print *,'############ After 2nd Syseqn in itnrml ############'
c                       pause
c                       pause
c                       pause
c                       print *,'neq=',neq
c                       print *,'disp(_) after 2nd syseqn solve =',(disp(ic),ic=1,ndof)
c                       pause
c                           print *,'b(i) after new 2nd syseqn=',(b(ic),ic=1,ndof)
c                           pause
c                       print *,'ad after 2nd syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                                   print *,'an after 2nd syseqn solvs=',(an(i),i=1,50)
c                       pause
       If(it .eq. 2)then
c      print *,'it = ',it
c                       print *,'disp after 2nd syseqn in itnrml=',(disp(ic),ic=1,ndof)
c                       print *,'b(_) after 2nd syseqn in itnrml=',(b(ic),ic=1,ndof)
c                       print *,'ad stiff after 2nd syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                       print *,'an(_) after 2nd syseqn in itnrml=',(an(ic),ic=1,50)
c         stop
         endif

c                       stop
c                       print *,'ae after 2nd syseqn=',(ae(i),i=1,maxdofpe**2)
c                       pause
c                       pause
c                       print *,'dqk1(_)=',(dqk1(ic),ic=1,9)
c                       pause
c** set "dqk1" cqual to the solution displ stored in "a"
       DO 178 LL=1,neq
cc       dqk1(LL) = a(LL)
       dqk1(LL) = disp(LL)
c                       print *,'dqk1(_) after new syseqn=',dqk1(LL)
 178   continue
c                       pause
c                       write(6,*)'dqk1(LL) inside itnml=',(dqk1(LL),i=1,neq)
c      print *,'a = ',a
c      print *,'dqk1 = ',dqk1
c      stop
c** new dqk1 from big "a"
cc      print *,'neq = ',neq
c       stop
c** get out "dqk1" as displacement, now put back into "a" at displace
c** NO leave dqk1 in own vector, donot put into "a"
c        do 177 ink=1,neq
c          dqk1(ink)=a(ink)
c c      print *,'a(ink) = ',a(ink)
c c      print *,'dqk1(ink) & ink = ',dqk1(ink),ink
c 177      continue
c      stop

c           call ludcmp(stifi,n,np,indx,X)
cc  c      print *,'n after ludcmp = ',n
c          stop
c           call lubksb(stifi,n,np,indx,dqk1)
c                       print *,'np=',np
c                       print *,'neq=',neq
c                       pause
c            do 12 i=1,np
             do 12 i=1,neq
 12            indx(i)=0

c** Now do the same for dqk2

c       call solve(dqk2)

c put dqk2(i) into big "a(i) as a force, get out as displacemnet"
        j=0
cc      print *,'neq, neq2, np2 = ',neq,neq2,np2
c        stop
c**********************************************
c       If(ityp .eq. 6)then
        do 32 i=1,ncq
c          j=j+1
c          a(i)=dqk2(j)
                          b(i)=dqk2(i)
c c      print *,'a(i) = ',a(i)
c       print *,'dqk2(i) = ',dqk2(i)
 32      continue
c                       print *,'#### After 3rd Syseqn inside itnrml ####'
c                       pause

c                       If(istep.gt.900)Then
c                       print *,'dqk2(_) inside itnml=',(dqk2(ic),ic=1,neq)
c                       print *,'b(_) inside itnml=',(b(ic),ic=1,ncq)
c                       pause
c                       Endif

c                       write(6,*)'dqk2(_) inside itnml=',(dqk2(ic),ic=1,neq)
c       Endif

c       If(ityp .eq. 12)then
c       do 33 i=ncq+1,ncq*3
c          j=j+1
c          a(i)=dqk2(j)
c c     print *,'a(i) = ',a(i)
c c     print *,'dqk2(j) = ',dqk2(j)
c 33     continue
c       Endif
c**********************************************

c        stop

cc      write(*,*) 'Before dcmpbd inside intml 2nd time'
c       stop
```

```
      ipick = 2

**********************


* Not sure if "a" is getting rewritten and used properly?????


**********************
cc      print *,'a before dcmpbd =',a
c       print *,'dqi before dcmpbd = ',dqi
c       print *,'dqk1(_) inside intrnl =',(dqk1(ic),ic=1,neq)
c                       pause
c       stop
cc      call dcmpbd
c     stop
c       write(*,*) 'After dcmpbd inside intrnl 2nd time'
cc      call slvbd
c       write(*,*) 'After slvbd & ipick=2 inside intrnl 2nd time'
c                       print *,'############### END SYSEQN 3 ###############'
c... I only want to solve, I don't want update forces or stiff
                          getforce=0
                          getstiff=0
                          getsolve=1
c                       print *,'getsolve=1 inside itnrml before 3rd syseqn'
c                       pause

c                       print *,'b(i) before new 3rd syseqn=',(b(ic),ic=1,ndof)
c                       pause
c                       print *,'ad before 3rd syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                       pause
c                       print *,'disp(_) before 3rd syseqn solve only=',(disp(ic),ic=1,ndof)
c                       pause
c...Need to solve eqns only, no new loads or stiff!
        call syseqn(nie,iplvl,b,ndof,disp,iarc,qk,
     $              ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $              neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $              iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dquin1,
     $              ux,uy,uz,di,it,area,youngm,
     $           ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $ zee,dLength,iex)
c                       print *,'############### END SYSEQN 3 ###############'
c                       pause
c                       print *,'neq=',neq

c                       print *,'b(i) after new 3rd syseqn=',(b(ic),ic=1,ndof)
c                       pause
c                       print *,'ad after 3rd syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                       pause
c                       print *,'an after 2nd syseqn solvs=',(an(i),i=1,50)
c                       stop

c         If(it .eq. 2)then
c       print *,'it = ',it
c                       print *,'disp after 3rd syseqn in itnrml=',(disp(ic),ic=1,ndof)
c                       print *,'b(_) after 3rd syseqn in itnrml=',(b(ic),ic=1,ndof)
c                       print *,'ad stiff after 3rd syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                       print *,'an(_) after 3rd syseqn in itnrml=',(an(ic),ic=1,50)
c         stop
c         endif

c                       print *,'disp(_) after 3rd syseqn solve only=',(disp(ic),ic=1,ndof)
c                       pause

c** set "dqk1" equal to the solution displ stored in "a"
        DO 179 LL=1,neq
cc        dqk2(LL) = a(LL)
        dqk2(LL) = disp(LL)
c       print *,'dqk2 = ',dqk2(LL)
 179    continue
cc      print *,'a after slvbd = ',a
c       print *,'dqk1 = ',dqk1
c       print *,'dqk2 = ',dqk2
c       stop
c           call stiff
ccccc             call syseqn(nc,iplvl)
c       stop
            do 13 i=1,np
 13           indx(i)=0

c ... compute dlk using eqn. 4.101
        dprd1 = 0.d0
        dprd2 = 0.d0
        do 40 i=1,neq
c                       print *,'dqi(_) dqk1(_) dqk2(_) = ',dqi(i),dqk1(i),dqk2(i)
          dprd1 = dprd1 + dqi(i)*dqk2(i)
          dprd2 = dprd2 + dqi(i)*dqk1(i)
 40       continue
c                       write(6,*)'dprd1 inside itnrml=',dprd1
c                       write(6,*)'dprd2 inside itnrml=',dprd2

c                       If(istep.gt.900)Then
c       print *,'dqi(i)=',dqi
c                       print *,'istep=',istep
c       print *,'dqk1(i)=',(dqk1(ic),ic=1,ndof)
c       print *,'dqk2(i)=',(dqk2(ic),ic=1,udof)
c                       print *,'dprd1=',dprd1
c                       print *,'dprd2=',dprd2
c                       print *,'ad(_)=',(ad(ic),ic=1,ndof)
c                       pause
c                       Endif

c                       Testv2=dlk*(dprd2+dlim1)
c                       print *,'istep=',istep
c                       print *,'Testv2=',testv2
c                       pause

c                       If(Testv2.lt.-1.6e-5)Then
c                       dli=-1.*dli
c                       Else
c                       dli=1.*dli
c                       Endif
c                       If(dprd1 .lt. 0.0 .and. dprd2 .lt. 0.0)Then
c                         If(dprd1 .gt. -.00001 .and. dprd2 .gt. -.1)Then
c                         dli=-1.*dli
c                         Endif
c                       Else
c                         dli=1.*dli
c       Endif

c       print *,'dprd1 = ',dprd1
c       print *,'dprd2 inside itnrml= ',dprd2
c                       pause
cc      print *,'dli insidc itnrml= ',dli
c       stop
c       stop
        dlk = (dprd1)/(dprd2 + dli)
c       print *,'dlk inside itnrml = ',dlk
cc      stop
c ... update the external load multiplier exlk
c       print *,'exlk before = ',exlk

c                                       If(istep .gt. 561)then
c                                       dlk = -1.d0*dlk
c                                       Endif

        exlk = exlk - dlk
c                                       If(istep .eq. 561)then
c                                       exlk = -1.d0*exlk
c                                       Endif


c       print *,'1st exlk in itnrml = ',exlk
c                       pause
c                       print *,'dlk in itnrml=',dlk
c                       pause
c       write(6,*)'1st exlk in itnrml = ',exlk
c                       write(6,*)'dlk in itnrml=',dlk
c                       stop

ccc     pause

c       stop
c ... update dlim1 for next load step
        dlim1 = dlim1 - dlk
c       stop
c ... compute dqk using eqn. 4.94
c ... update dqim1 for next load step
c ... update the total displacements using eqn. 4.103
c       print *,'neq = ',neq
c       stop
        do 50 i=1,neq
c                       If(istep.eq.914)Then
c         dqk(i) = dqk2(i) + dlk*dqk1(i)
c       Else
          dqk(i) = dqk2(i) - dlk*dqk1(i)
c                       Endif
c                       If(dqk(11).lt.0)Then
c                       print *,'istep=',istep
c       print *,'dqk(11)=',dqk(11)
c                       pause
c                       Endif
c         stop
c         dlim1(i) = dlim1(i) + dqk(i)
          dqim1(i) = dqim1(i) + dqk(i)
c       print *,'dqim1(i) & i = ',dqim1(i),i
c                       print *,'qk(11)before=',qk(11)
          qk(i) = qk(i) + dqk(i) !Update Total Displacements
c       print *,'qk(11) after=',qk(11)
 50     continue

c                       print *,'istep=',istep
c       print *,'dqk(11)=',dqk(11)
c       print *,'qk(11)=',qk(11)
c                       pause

c                       If(istep.gt.900)Then
c                       print *,'istep=',istep
c                       print *,'dqi(_)=',(dqi(ic),ic=1,neq)
c                       print *,'dqk(_)=',(dqk(ic),ic=1,neq)
c                       print *,'qk(_)=',(qk(ic),ic=1,neq)
c                       print *,'dli(_)=',(dli(ic),ic=1,neq)
c                       print *,'dlk(_)=',(dlk(ic),ic=1,neq)
c                       print *,'ad(_)=',(ad(ic),ic=1,neq)
c                        pause
c       endif
c         Do 5050 ii=1,neq
c           dqim1(ii) = dqim1(ii) + dqk(ii)
c5050                   continue
cc      If(istep.gt.911)Then
c       If(istep.gt.136)Then
cc                      print *,'istep=',istep
cc                      print *,'dqk1(i)=',(dqk1(ic),ic=1,neq)
cc                      print *,'dlk=',dlk
cc                      print *,'dqk2(i)=',(dqk2(ic),ic=1,neq)
cc      print *,'dqk(i)=',(dqk(ic),ic=1,neq)
cc                      print *,'qk(i)=',(qk(ic),ic=1,neq)
cc                      print *,'ad(ic)=',(ad(ic),ic=1,neq)
cc                      pause
cc                      Endif
c                       write(6,*)'qk(i)=',(qk(i),i=1,neq)
c                       print *,'qk(i)=',(qk(ic),ic=1,neq)
c                       pause
c       stop

c       print *,'exlk =',exlk
cc      print *,'dlim1 =',dlim1
cc      print *,'dqk=',dqk
cc      print *,'qk=',qk
cc      print *,'dqim1(1)=',dqim1
c ... compute the internal forces, fintk, at qk
c       call forces
```

```
c     print *,'Check inside itnrml before syseqn'
c       stop
c put qk(i) into big "a(i) as a disp, get out fintk"
      j=0
cc     print *,'neq, neq2, np2 = ',neq,neq2,np2
c       stop
       do 51 j=1,neq
cc          a(j)=qk(j)
                        disp(j)=qk(j)
c c     print *,'a(j) = ',a(j)
c c     print *,'qk(j) = ',qk(j)
 51     continue
c       stop
                        getsolve = 0
       getforce = 1
       getstiff = 0
c      print *,'fintk = ',fintk
c       stop
c      write(6,*)'Check inside itnrml before 4th syseqn'

c                       print *,'b(i) before 4th syseqn=',(b(ic),ic=1,ndof)
c                       print *,'ad before 4th syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                       print *,'an before 4th syseqn solvs=',(an(i),i=1,50)
c                       print *,'disp before 4th syseqn in itnrml=',(disp(ic),ic=1,neq)
c                       stop

c                       print *,'Check inside itnrml before 4th syseqn'
c                       print *,'!!!!!!!!!!!!!! 4th syseqn inside itnrml !!!!!!!!!!!!!!!'
c                       pause
c                       stop
       call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $            ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $            neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $            iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqiml,
     $            ux,uy,uz,di,it,area,youngm,
     $         ianal,lumpmass,iskip,itempo2,itnrmlon,cxfor,istep,
     $ zee,dLength,icx)

c     $ja,iboundc,ndofped,bc,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c      print *,'getforce = ',getforce
c      write(6,*)'Check inside itnrml after 4th syseqn'
c      print *,'Check inside itnrml after 4th syseqn'
c                       pause
c                       print *,'ad(_) in itnrml after 4th syseqn=',(ad(ic),ic=1,neq)
c                       pause

c^^^^^ SETTING ZERO DIAGONAL VALUES TO ONE ^^^^^^^^^^^^^^^^
c       Do 1015 i=neq*2+1,neq*3
                          Do 1015 i=1,neq
c       If(ad(i) .GT. .000001) GoTo 1015
       If(ad(i) .GT. 1.e-8) GoTo 1015
c         a(i)=1.0
c                          ad(i)=1.e-8
c                          ad(i)=0.d0
1015    Continue
c                       print *,'b(i) after new 2nd syseqn=',(b(ic),ic=1,ndof)
c                       print *,'ad after 2nd syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                       print *,'an after 2nd syseqn solvs=',(an(i),i=1,50)
c                       print *,'disp after 4th syseqn in itnrml=',(disp(ic),ic=1,neq)
c                       stop

c                       print *,'qk after 4th syseqn in itnrml=',(qk(ic),ic=1,20)
c                       print *,'dqi after 4th syseqn in itnrml=',(dqi(ic),ic=1,neq)
c                       pause

         do 703 i=1,neq
c              scali(i) = 1.d0/stifi(i,i)
c     c c      print *,'a(neq*2+i) = ',a(neq*2+i)
cc               scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
               scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c                       print *,'scali =',scali(i)
 703        continue
c                       pause

c                       write(6,*)'scali after 4th syseqn itnrml=',(scali(i),i=1,neq)
c                       print *,'scali after 4th syseqn itnrml=',(scali(i),i=1,neq)
c c     print *,'a=',a
c         stop
cc          call modify
c      print *,'Check after sixth modify'
c c     print *,' "a" after 6th modify = ',a
c      c     pause
         getforce=0 !set back to zero
c c     print *,' "a" after 3rd syseqn to getforce = ',a
c       stop
c** The DO below looks at dqk and assigns 0 to correspond
c** to the boundary conditions which were zeroed out in "a"
c** so if the displacements "qk" stored in the beginning of
c** big "a" equal zero, then "dqk(i)" must be zero.
       Do 49 i=1,neq
c       If(a(i).eq. 0.0)then
cc        If(disp(i).eq. 0.0)then
                            If(iboundc(i) .eq. 1)then
       dqk(i)=0.d0
       endif
c      print *,'dqk(_)=',dqk(i)
 49     continue

cc     print *,'fintk 1a = ',fintk
c       stop

c        If(it .eq. 2)then
cc     print *,'it = ',it
cc     print *,'dqk(i) = ',dqk
c         stop
c         endif
c      print *,'a before resid = ',a
c      stop
c                       print *,'***chk before resid end itnrml***'
c                       pause
c      print *,'exlk before resid=',exlk

c      print *,'exfor before resid=',(exfor(i),i=1,neq)
c      print *,'b before resid=',(b(i),i=1,neq)
c                       pause
c ... compute the residual forces, resk, at qk and exlk
       call resid(dqtot,dqi,dqiml,qk,b,exfor)
c        print *,'a after resid = ',a
c        stop
c      print *,'resk(_) after resid= ',(resk(ic),ic=1,neq)
c      print *,'exlk after resid= ',exlk

c                       print *,'disp after resid in itnrml=',(disp(ic),ic=1,neq)
c                       print *,'dqtot after 4th syseqn in itnrml=',(dqtot(ic),ic=1,neq)

        If(it .eq. 2)then
c      print *,'it = ',it
c                       print *,'disp end itnrml=',(disp(ic),ic=1,ndof)
c                       print *,'b(_) end itnrml=',(b(ic),ic=1,ndof)
c                       print *,'ad end itnrml=',(ad(ic),ic=1,maxdofpe**2)
c                       print *,'an(_) end itnrml=',(an(ic),ic=1,50)
c         stop
          endif

c                       pause
c         If(it .eq. 2)then
c      print *,'resk(_) = ',(resk(ic),ic=1,neq)
c      print *,'exlk = ',exlk
c                       pause
c         stop
c         endif
c      print *,'stop after resid'
c       stop
c ... test for convergence
cc      print *,'toldis tolfor exlk iconv=',toldis,tolfor,exlk,iconv
c       call test(toldis,tolfor,exlk,iconv)
c                       print *,'chk before test at end itnrml'
c                       print *,'istep=',istep
c                       pause
       call test(dqtot,dqi,dqiml,qk,exfor,b,itnrmlon)
c        call test(disp,dqi,dqiml,qk,exfor)
c      print *, 'iflag after test = ',iflag
c      print *,'iconv after test = ',iconv

        If(it .eq. 2)then
c      print *,'it = ',it
c                       print *,'iconv when it=2 =',iconv
c                       print *,'iflag when it=2 =',iflag
c         stop
          endif

c                       pause
c      print *,'stop after test'
c      print *,'a at end of itnrml = ',a

c                       print *,'b(i) end itnrml=',(b(ic),ic=1,ndof)
c                       pause
c                       print *,'ad end itnrml=',(ad(i),i=1,maxdofpe**2)
c                       print *,'an end itnrml=',(an(i),i=1,50)
c                       pause
c                       print *,'disp end itnrml=',(disp(ic),ic=1,neq)
c                       pause
c                       stop
c** Initate a counter for going back to 1002
        iam=iam+1
cc      print *,'iam at end=',iam
c      print *,'exlk at end = ',exlk
c       stop
c       print *,'iconv =',iconv
c       Print *,'it = ',it
c       print *,'itemax =',itemax
c       pause
        go to 1002
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      elseif (iconv .ne. 0 .and. it .gt. itemax) then
c      print *,'it = ',it
c      print *,'Check at iflag = 2'
c       print *,'iconv =',iconv
c       Print *,'it at end of itnrml = ',it
c       print *,'itemax =',itemax
       iflag = 2
      end if
c      print *,'exlk at end of itnrml = ',exlk
c      print *,'Ready to leave itnrml'
c      stop
c
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine funct(iexample,u,f,fd,neq,iforce,L1,L1i)
      implicit real*8(a-h,o-z)
      dimension u(*),f(*),fd(2)
      double precision L1,L1i,hgt,Q,R,S,T,V,sprk,f,fd
c
      go to (11,12,13,14,15,16,17,18,19),iexample
c.....Duc's automatic large scale nonlinear exaanples
c.....correct (exact) solution = {1, 1, 1, ..., 1}
 11    continue
      do 21 i=1,neq
      f(i)=0.0
      rhs=0.0
       do 22 j=1,neq
       coef=i+j
       if (i .eq. j) coef=coef*(10.0*neq)
       termij=coef*u(i)*u(j)*u(j)
       f(i)=f(i)+termij
       rhs=rhs+coef
 22    continue
      f(i)=f(i)-rhs
 21    continue
      go to 999
c.....Arora's nonlinear example, page 336
c.....correct (exact) solution = {-.397975,0.540417}
```

```
 12   continue
      f(1)=3.d0*u(1)**2+12.d0*u(2)**2+10.d0*u(1)
      f(2)=24.d0*u(1)*u(2)+4.d0*u(2)+3.d0
      print *,'f(1) inside funct = ',f(1)
      print *,'f(2) inside funct = ',f(2)
      pause
      go to 999
c.....Arora's nonlinear example, page 331
c.....Minimize phi = 5X1**2+2X1X2+X2**2+7
c.....correct (exact) solution = {0, 0}
 13   continue
      f(1)=10.d0*u(1)+2.d0*u(2)
      f(2)=2.d0*u(1)+2.d0*u(2)
      go to 999
c.....Arora's nonlinear example, page ???
c.....correct (exact) solution = {-1/3, -1/2}
 14   continue
      f(1)=24.d0*u(1)-12.d0*u(2)+2.d0
      f(2)=8.d0*u(2)-12.d0*u(1)
      go to 999
 15   continue
c.....J. Dennis's optimization book, page 172
c.....(hard time for BFGS ??) example
c.....correct (exact) solution = {0,3}, and also {3.0}
      f(1)=u(1)+u(2)-3.0
      f(2)=u(1)**2+u(2)**2-9.0
      go to 999
 16   continue
c.....J. Dennis's optimization book, page 201
c.....correct exact) solution = {2,-1}
      f(1)=4.d0*(u(1)-2.d0)**3+2.d0*(u(1)-2.d0)*u(2)**2
      f(2)=2.d0*(u(1)-2.d0)**2*u(2)+2.d0*(u(2)+1.d0)
      go to 999
 17   continue
c.....J. Dennis's optimization book, page ???
c.....correct exact) solution = {0.00228,0.00228}
      f(1)=4.d0*u(1)**3+4.d0*u(1)*u(2)**2
      f(2)=4.d0*u(2)*u(1)**2+4.d0*u(2)**3
      go to 999
 18   continue
c.....Osman's nonlinear hydraulic example
c.....correct solution = {-10.0, 40.0, 30.0, 157.1}
      c1=-0.02057
      f(1)=c1*u(1)*dabs(u(1)) - u(4) + 155.0
      f(2)=c1*u(2)*dabs(u(2)) - u(4) + 190.0
      f(3)=c1*u(3)*dabs(u(3)) + u(4) - 138.6
      f(4)=u(1)+u(2)-u(3)
      go to 999
c.....Mathcad Indefinite case
c.....correct (exact) solution = {1.089,.839}
c19    continue
c     f(1)=u(1)+u(1)*u(2)-2.d0
c     f(2)=u(1)**2+4.d0*u(2)**2-4.d0
c     go to 999
c.....Nonlinear Truss example
19    continue
      f(1)=0.d0
      A = 1.d0
      E = 1.d7
          sprk = 3.d0
          hgt = 1.d0
      print *,'A, E, sprk, hgt inside funct =',A,E,sprk,hgt
       print *,'fd(1) inside funct = ',fd(1)
c      stop
c     L1 = dsqrt((100.d0)**2 + (hgt)**2)
c     L1i = dsqrt((100.d0)**2 + (hgt-u1)**2)
c     L = 10.d0
c     L1=14.1421d0
c     L2=10.d0
c     L1i=0.d0
c     L2i=0.d0
c     D = 14.142d0

c     u2 = u(2)
c     print *,'u1 = ',u1
c     print *,'u2 = ',u2
c     pause
c      L1i = dsqrt((10.d0+u1)**2 + (L2+u2)**2)
c      print *,'L1i = ',L1i
c      L2i = dsqrt((L2+u2)**2 + u1**2)
c      print *,'L2i = ',L2i
c      pause
c      Rcos1 = (10.d0+u1)/L1i
c      Rcos1a = 10.d0/L1
c      Rsin1 = (L2 + u2)/L1i
c      Rsin1a = L2/L1
c      Rcos2 = u1/L2i
c      Rcos2a = 0.d0
c      Rsin2 = (L2 + u2)/L2i
c      Rsin2a = 1.d0

c       P1 = (A*E/L1) * (L1i-L1)
c       P2 = (A*E/L2) * (L2i-L2)
c      print *,'P1 = ',P1
cc       print *,'P2 = ',P2
c      pause
c      print *,'u1 inside funct = ',u1
c      print *,'u2 inside funct = ',u2
c      print *,'fd(1) & fd(2)',fd(1),fd(2)
c...u1 is the incremental displacements from previous step
      u1 = u(1)
      print *,'u1 inside funct = ',u1
      print *,'iforce = ',iforce
c     stop
c     If(iforce.eq.1)then
      L1 = dsqrt((100.d0)**2 + (hgt)**2)
c     Else
c...The new deformed geometry ready for next step
c      L1 = dsqrt((100.d0)**2.d0 + (hgt-u1)**2.d0)
      print *,'New starting length L1 inside funct = ',L1
c...get next guess using old u1 plus .1
c       u1=u1+.1

c     Endif
      L1i = dsqrt((100.d0)**2.d0 + (hgt-u1)**2.d0)
      print *,'hgt inside funct = ',hgt
      print *,'L1 inside funct = ',L1
      print *,'L1i inside funct = ',L1i
      print *,'fd(1) inside funct = ',fd(1)
c     stop
c     pause
c     print *,'A, E, L1 inside Stiff',A,E,L1
c Below stifi is straight stiff from VPI desertation
c      f(1)=((A*E/L1)*((hgt-u1)/L1i)**2.d0 +((A*E/L1i)*(L1i-L1))/L1*
c     &      (1.d0 - ((hgt-u1)/L1i)**2.d0) + sprk)*u1-fd(1)
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c...This is the internal force
c       forci=A*E*(1.d0/L1i - 1.d0/L1)*(hgt-u1)+sprk*u1
c       print *,'forci inside funct = ',forci
c Below is Jay's sdof problem minus forci
ccc       f(1)=( (A*E/L1)*((hgt-u1)/L1i)**2 + (A*E/L1i)*
ccc     &    (L1i-L1)/L1 * (1.d0 - ((hgt-u1)/L1i)**2)+sprk)*u1-fd(1)
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c       f(1)=( (A*E/L1)*((hgt-u1)/L1i)**2 + (A*E/L1i)*
c     &    (L1i-L1)/L1 * (1.d0 - ((hgt-u1)/L1i)**2)+sprk)*u1-fd(1)

c...Try, "func=K*u - force"
c     f(1) = ( A*E*( (1.d0/L1) - (1.d0/L1i) +
c     & ( (1.d0 - u1)**2.d0/L1i**3.d0) )+sprk )*u1-fd(1)

      f(1) = ( A*E*( (1.d0/L1) - (1.d0/L1i) +
c     & ( (hgt**2 - 2.d0*hgt*u1 + u1**2)/L1i**3) )+sprk )*u1-forci
     & ( (hgt**2 - 2.d0*hgt*u1 + u1**2)/L1i**3) )+sprk )*u1-fd(1)
      print *,'f(1) inside funct = ',f(1)
c     stop
c     pause

c      f(1) = ((A*E/L1) * .5d0 +
c     &       (A*E/L2)*(dsqrt((L2+u2)**2+u1**2)-L2)/L2)*u1 +
c     &        ((A*E/L1)*.5d0)*u2 - 0.d0
c      f(2) = ((A*E/L1)*.5d0)*u1 +
c     &        ((A*E/L1)*.5d0 + (A*E/L2))*u2 - fd(2)
cc     f(1) = ((A*E/L1) * Rcos1**2 + P1*(1-Rcos1**2)/L1i +
cc     &      (A*E/L2) * Rcos2**2 + P2*(1-Rcos2**2)/L2i) * u1 +

cc     &        ((A*E/L1)*Rsin1*Rcos1 + P1*(-Rcos1*Rsin1)/L1i +
cc     &         (A*E/L2)*Rsin2*Rcos2 + P2*(-Rcos2*Rsin2)/L2i) * u2

cc     f(2)=  ((A*E/L1)*Rsin1*Rcos1 + P1*(-Rcos1*Rsin1)/L1i +
cc     &      (A*E/L2)*Rsin2*Rcos2 + P2*(-Rcos2*Rsin2)/L2i) * u1 +

cc     &         ((A*E/L1)*Rsin1**2 + P1*(1-Rsin1**2)/L1i +
cc     &          (A*E/L2)*Rsin2**2 + P2*(1-Rsin2**2)/L2i) * u2

c******** Linear Case *********
cx      f(1) = ((A*E/L1) * .5d0)*u1 +
cx     &        ((A*E/L1)*.5d0)*u2 - 0.d0
cx      f(2) = ((A*E/L1)*.5d0)*u1 +
cx     &        ((A*E/L1)*.5d0 + (A*E/L2))*u2 - 2.d6
c
 999  continue
c     write(6,*) 'current sol = ',(u(i),i=1,neq)
c     write(6,*) 'current {f} inside funct = ',(f(i),i=1,neq)
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.2 Hrinda Rikarc Optimization

```
      implicit real*8(a-h,o-z)
      dimension nel(6),ndofpe(6),iflag(100)  ! assuming 6 finite element types
                dimension dqi(5000),dqk(5000)

     common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
     common/disp/ux(2),uy(2),uz(2),P(6),B1(6)

                common/RIK/ stifi(60,60),dqk1(5000),
     $    dqk2(5000),fintk(5000),
     $    resk(5000),scali(5000),work1(5000),indx(5000),neq

                common/testing/toldis,tolfor,exlk,iconv

      integer, allocatable:: ie(:),je(:),iboundc(:),iet(:),jet(:)
     $,ip(:),iu(:),iut(:),iup(:),isupn(:),lm(:),iperm(:),ia(:),ja(:)
     $,ju(:),jut(:),itempo1(:),iarea(:),ilimit(:),kprop(:),
     $ iSEDchk(:),minarea(:)

      double precision, allocatable:: b(:),be(:),ae(:),ad(:)
     $,x(:),y(:),z(:),propmat(:),propsect(:),dqtot(:)
     $,tempo1(:),tempo2(:)
     $,di(:),an(:),un(:),evalues(:),evectors(:),dm(:),am(:)
     $,elm(:,:)
     $,disp(:),qk(:),exfor(:)
     $,strainel(:),strainE(:),strainD(:),weight(:),Anew(:)
     $,Anew2(:),Ancw3(:),VED(:),VEDold(:),Apart(:),SEDsum(:)
     $,Fin1(:),Fin2(:),WF(:),Dlimit(:),xlie(:),SED(:)

      double precision, allocatable:: tempo3(:)
c
c                 open(unit=5,file='input3.dat',status='old')
c                 open(unit=6,file='output3.dat',status='old')

cbar              open(unit=5,file='bar2.dat',status='old')
carch             open(unit=5,file='arch3din.dat',status='old')
carch2            open(unit=5,file='arch2in.dat',status='old')
```

```
carch3          open(unit=5,file='arch3in.dat',status='old')
cframe          open(unit=5,file='frame_in.dat',status='old')
cframe2         open(unit=5,file='frame2_in.dat',status='old')

ct3a            open(unit=5,file='t3a_in.dat',status='old')!3mem truss triangle
cpod            open(unit=5,file='tpod_in.dat',status='old')
c               open(unit=5,file='tpod2_in.dat',status='old')
ctwo            open(unit=5,file='twobar_in.dat',status='old')

csdof           open(unit=5,file='cris_sdof_in.dat',status='old')
ckhot           open(unit=5,file='khot_2ele_in.dat',status='old')

                open(unit=5,file='khot_2ele_mod_in_unsym.dat',status='old')
                open(unit=8,file='khot_2ele_mod_in_unsym_areas.dat',status='old')

ckhot2_sym      open(unit=5,file='khot_2elc_mod_in_sym.dat',status='old')
ckhot2_sym      open(unit=8,file='khot_2clc_mod_in_sym_areas.dat',
ckhot2_sym    c   status='old')

ckhot4          open(unit=5,file='khot_4ele_mod_in.dat',status='old')
ckhot4          open(unit=8,file='khot_4elc_mod_in_areas.dat',status='old')

cdome           open(unit=5,file='dome2_opt_mod_in.dat',status='old')
cdome           open(unit=8,file='dome2_opt_mod_in_areas.dat',status='old')

cshallow        open(unit=5,file='shallow_truss_in.dat',status='old')
cshallow        open(unit=8,file='shallow_truss_areas.dat',status='old')

                open(4,file='SED_out.txt',status='unknown')
c               open(3,file='SED_out.txt',status='unknown')
c               open(2,file='SED.txt',status='unknown')

c               open(unit=5,file='khot_6ele_mod_in.dat',status='old')
c               open(unit=5,file='khot_10ele_mod_in.dat',status='old')
c               open(unit=5,file='khot_11ele_mod_in.dat',status='old')
c               open(unit=5,file='khot_12elc_mod_in.dat',status='old')
c               open(unit=5,file='khot_18ele_mod_in.dat',status='old')
c               open(unit=5,file='khot_20ele_mod_in.dat',status='old')

cxxx            open(unit=5,file='dome2_opt_mod_in.dat',status='old')

cdome2          open(unit=5,file='dome2_opt_mod_in_initial.dat',status='old')

carch3d         open(unit=5,file='arch3d_opt_mod_in_initial.dat',status='old')

carch3d_half    open(unit=5,file='arch3d_opt_mod_half_in_initial.dat',status='old')

carch41         open(unit=5,file='arch41_opt_mod_half_in_initial.dat',status='old')

c               open(unit=5,file='dome_in.dat',status='old')

csnap           open(unit=5,file='snap_in.dat',status='old')

csnap4          open(unit=5,file='snap4_in.dat',status='old')

c               open(unit=5,file='snap5_in.dat',status='old')


cdome7          open(unit=5,file='dome7_in.dat',status='old')
ctruss          open(unit=5,file='truss7_3d_in.dat',status='old')

c               open(unit=5,file='bar2a.dat',status='old')
c               open(unit=5,file='bar2b.dat',status='old')
cbar            open(unit=6,file='bar2out.dat',status='old')
carch           open(unit=6,file='arch3dot.dat',status='old')
cframe          open(unit=6,file='frame_out.dat',status='old')
ctwo            open(unit=6,file='twobar_out.dat',status='old')

csdof           open(unit=6,file='cris_sdof_out.dat',status='old')
ckhot           open(unit=6,file='khot_2elc_out.dat',status='old')
cdome2          open(unit=6,file='dome2_opt_out.dat',status='old')
cdome           open(unit=6,file='dome_out.dat',status='old')

csnap           open(unit=6,file='snap_out.dat',status='old')
ct3a            open(unit=6,file='t3a_ot.dat',status='old')
cpod            open(unit=6,file='tpod_ot.dat',status='old')
ctruss          open(unit=6,file='truss7_3d_ot.dat',status='old')
                open(unit=7,file='data_out.dat',status='old')
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
c
c     write(6,*) 'Version Date = 03-01-2004'
c.....input general information
      neltype=6               ! assuming 6 finite element types
                ityp=12 ! type ele is 3d from GAH
                iopt=0 !if=1 then going thru rik again for optimizer!
                N=1 !counter for times thru rikarc

c...area ratio=1 means section areas have been ratioed after rikarc
                iratio=0 !area ratio
                weightotal=0.d0
                weightold=0.d0


c...below used in determining (SSP)step size parameter
                iexlk=0!this is 1 if exlkdl0, 2 if exlkdli & 3 if exlkdlk

c               allocate(VED(100),VEDold(100),iarea(100))!100 = num elements

1000            Continue !Dummy statement
                rewind (5)
c               open(unit=4,file='dome2_opt_mod_in_unit.dat',status='old')

cdome2          open(unit=5,file='dome2_opt_mod_in2.dat',status='old')

carch3d         open(unit=5,file='arch3d_opt_mod_in2.dat',status='old')

carch3d_half    open(unit=5,file='arch3d_opt_mod_half_in2.dat',status='old')
carch41         open(unit=5,file='arch41_opt_mod_half_in2.dat',status='old')

cshallow        open(unit=5,file='shallow_truss_in2.dat',status='old')
```

```
ckhot2_sym      open(unit=5,file='khot_2ele_mod_in_sym.dat',status='old')
                open(unit=5,file='khot_2ele_mod_in_unsym.dat',status='old')
ckhot4          open(unit=5,file='khot_4ele_mod_in.dat',status='old')
cdome           open(unit=5,file='dome2_opt_mod_in.dat',status='old')
cshallow        open(unit=5,file='shallow_truss_in.dat',status='old')

c               open(unit=5,file='dome2_opt_mod_in_applied.dat',status='old')
c               Endif

                print *,'check start N= ',N
     call generalip(nel,ndofpc,neltype,maxdofpe,nboundc
    $,          ndofpn,numnodes,loadof,nummat,nlayers,nsect
    $,          lumpmass,ndof,ieall,jeall,ianal,neig,iarc
    $,          itype,numlim)


                inumel=nel(1)

ccc             iopt=0 !if=1 then going thru rik again for optimizer!
ccc             N=1 !counter for times thru rikarc

c.....dynamic storage memory allocation
      allocate ( ie(ieall+1),je(jeall),iboundc(ndof),iet(ndof+1),
c..changed ip(ndof) to ip(ndof+1)
    $  jet(jeall),ip(ndof+1),iu(ndof+1),iut(ndof+1),iup(ndof),
    $  isupn(ndof),lm(maxdofpe),iperm(ndof),ia(ndof+1))

                 allocate (ja(ndof+1))

      allocate ( b(ndof),be(maxdofpe),ae(maxdofpe**2),ad(ndof),
    $     an(10000),dqtot(ndof),
    $     x(numnodes),y(numnodes),z(numnodes),
    $     propmat(nummat*11),tempo1(ndof),
    $     tempo2(ndof),di(ndof),exfor(ndof))

      allocate ( disp(ndof),qk(ndof) )

                allocate (strainel(ndof),strainE(ndof),strainD(ndof),
    $       weight(ndof))

                 allocate (Anew(ndof),Anew2(ndof),Anew3(ndof),
    $ Apart(ndof),SEDsum(ndof),Fin1(ndof),Fin2(ndof))
ccccccccccccccccc
                If(N .eq. 1)then
                allocate(VED(ndof),VEDold(ndof),iarea(inumel),propsect(nsect))
                allocate(minarea(nsect))
                allocate(WF(inumel),ilimit(ndof),dlimit(ndof),kprop(inumel))
                allocate(xlic(inumel))
                allocate(SED(inumel))
                allocate(iSEDchk(inumel))
                Endif
ccccccccccccccccc
c.....input material properties

     call materprop(nummat,propmat,youngm)

c.....input cross-sectional properties
                print *,'N before sectprop=',N
                If(N .eq. 1)then
     call sectprop(nsect,propsect,area)
c...set to zero, after rikarc is run, minarea may change to 1
c...minarea is either 0 or 1, when it is 1, means that the min
c...arca value has been reached in the design and the area may
c...not go below this value.
                 Do 444 i=1,nsect
444               minarea(i)=0

                Endif

c.....input joint coordinates
     call nodecoor(numnodes,x,y,z)

c.....input connectivity information
     call elconect(ie,je,lm,neltype,nel,ndofpc,ndofpn,iarca)
c.....input applied loads at the joints
     call loads(ndof,b,loadof,exfor)
c               print *,'exfor =',(exfor(i),i=1,ndof)

                call limits(ndof,numlim,dlimit,ilimit)
c               print *,'b after loads=',(b(i),i=1,ndof)
c.....input boundary dof
                print *,'chk1 before supportdof'

cc              iopt=0 !if=1 then going thru rik again for optimizer!
cc              N=1 !counter for times thru rikarc
                AEDsum=0.d0 !sum strain energy density, set to zero

                Do 555 j=1,inumel
 555            SEDsum(j)=0.d0

1100            Continue
c               print *,'chk2 before supportdof'
c               open(unit=51,file='dome_sup.dat',status='old')

carch3d         open(unit=51,file='arch3d_opt_mod_sup.dat',status='old')

carch3d_half    open(unit=51,file='arch3d_opt_mod_half_sup.dat',status='old')

carch41         open(unit=51,file='arch41_opt_mod_half_sup.dat',status='old')

ckhot_sym       open(unit=51,file='khot_2ele_mod_sym_sup.dat',status='old')
                open(unit=51,file='khot_2elc_mod_unsym_sup.dat',status='old')

ckhot4          open(unit=51,file='khot_4elc_mod_sup.dat',status='old')

cdome           open(unit=51,file='dome2_sup.dat',status='old')

cshallow        open(unit=51,file='shallow_truss_sup.dat',status='old')

c               open(unit=51,file='sdof_sup.dat',status='old')
c               open(unit=51,file='Khot_4ele_sup.dat',status='old')
c               open(unit=51,file='Khot_6ele_sup.dat',status='old')
c               open(unit=51,file='Khot_10ele_sup.dat',status='old')
```

```
c                 open(unit=51,file='Khot_11ele_sup.dat',status='old')
c                 open(unit=51,file='Khot_12ele_sup.dat',status='old')
c                 open(unit=51,file='Khot_18ele_sup.dat',status='old')
c                 open(unit=51,file='Khot_20ele_sup.dat',status='old')

      call supportdof(iboundc,nboundc,b,ndof,ia)

c**Arc Length**
                  if(iarc .eq. 1)then
c                 print *,'<<<<<<<<<<<call to rikarc>>>>>>>>>>>'
                  call rikarc(ie,je,lm,neltype,ncl,ndofpe,ndofpn,
     $ numnodes,x,y,z,dqtot,
     $ ndof,b,loadof,
     $ iboundc,nboundc,ia,
     $ ieall,ji,ict,jct,
     $ itempo1,exfor,
     $ ja,
     $ ip,ad,an,propmat,propsect,be, !call assembly
     $ ae,maxdofpe,
     $ ianal,lumpmass,dm,am,elm,tempo1,
     $ iperm,  !call minfills
     $ in,ncoef2,ju, !call symfactd
     $ iu,iut,  !call transad
     $ isupn, !call supernode
     $ di,un,iup,iopf,  !call numfa1d
     $ iopb,  !call fbcd
     $ disp,iarc,jeall,ux,uy,
     $ qk,iarea,
     $ strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $ Fin2,VED,VEDold,xlie,exlkdl0,exlkdli,exlkdlk,SED,iexlk,
     $ weightotal,Nevenodd,N)
                  endif

c     Nevenodd = 2

                  dSEDtol=1.d-2 !Khot_4ele & 2ele
c                 dSEDtol=1.d0 !dome
c                 dSEDtol=1.d-1 !shallow truss
c                 dSEDtol=5.0

                  Amin=.1d0
                  SEDtotal=0.d0
                  SEDtotal2=0.d0

                  Do 8 k=1,inumel
                  kprop(k)=0
 8                continue

                  Do 23 i=1,inumel
                  iSEDchk(i)=0 !set back to zero, if 1 then SED for ele within tol.
23    Continue

                  print *,'minarea(_) after rikarc=',minarea
c                 pause

                  print *,'propsect(_) at start=',propsect
c                 print *,'iSEDchk(_) at start=',iSEDchk
                  Do 22 i=1,inumel
22                print *,'SED(i) at start=',i,SED(i)
c                 pause

                   RN=N
c...N is odd, doing unit load
c...N is even, doing applied load
                  If(N.eq.1)then
                  Nevenodd = 1
                  goto 15
                  Endif

                  If(N/2 .eq. 2*RN/4)then
                   print *,'N is even'
                   Nevenodd = 2 ! This means even number.
                  Else
                   print *,'N is odd'
                   Nevenodd = 1 !This means odd number.
                  endif
15                Continue !dummy statement
c.....
c                  If(iratio .eq. 1)then ! we do this once.
c                  print *,'iratio = 1 .. STOP'
c                  stop
c                  Endif

                  sum=0.0
                  SEDavg=0.0
                    Do 95 i=1,inumel
                    sum=sum+SED(i)
95                  Continue
                  SEDavg=sum/inumel
                  Do 90 i=1,inumel
                                  iprop=iarea(i)
c                 print *,'iprop=',iprop
                  write(4,40) i,SED(i),propsect(iprop),exlk,SSP,SEDavg,N
c                 write(15,40) i,SED(i),propsect(iprop),exlk,SSP,SEDavg
40                format(i3,'SED=',e14.6,' Area = ',e14.6,' exlk= ',e14.6,
     $     ' SSP= ',e14.6,' SEDavg= ',e14.6,' N= ',i3)
90                Continue
c                 pause

                  write(4,*)'_____'
c...Checking the strain energy density (SED) difference between
c...all members, this is called dSED and is checked to see if it
c...is below the user defined tolerance called dSEDtol.
                  Do 360 i=1,inumel
                    Do 370 j=1,inumel
c                 print *,'i= j=',i,j

                  If(i .eq. j)then
                  iSEDchk(i)=1
                  print *,'goto 370'
                  goto 370 !skip, same element so dSED=1.
                  endif

                  print *,'iarea(i)=',iarea(i)
                  print *,'minarea(iarea(i))=',minarea(iarea(i))
                  print *,'iarea(j)=',iarea(j)
                  print *,'minarea(iarea(j))=',minarea(iarea(j))

                   If(minarea(iarea(i)) .eq. 1)Then
                   print *,'min area is being used for this member','i=',i,'j=',j
                   print *,'do not compare SED with other members'
c...set iSEDchk to 1 which means that the Element SED is below the
c...delta SED tolerance or min area has been reached in the design
                   iSEDchk(i)=1
      goto 360
                  Endif
                   If(minarea(iarea(j)) .eq. 1 )Then
                   print *,'min area is being used for this member','i=',i,'j=',j
                   print *,'do not compare SED with other members'
c...set iSEDchk to 1 which means that the Element SED is below the
c...delta SED tolerance or min area has been reached in the design
                   iSEDchk(j)=1
      goto 370
                  Endif
                    dSED=abs(SED(i)-SED(j))        !dSED is the delta SED between elements
                    print *,'SED(i)=',SED(i),' SED(j)=',SED(j)
                    print *,'i=',i,' j=',j,' dSED chk=',dSED

                  print *,'dSED=',dSED
                  print *,'dSEDtol=',dSEDtol
c                 pause

c..If the total array, dSED is filled with 1, this means it
c..is time to ratio the areas by exlk.
                   If(dSED .LT. dSEDtol) Then
                    iSEDchk(i)=1 !delta SED between two elements are below tolerance
                    print *,'iSEDchk(_)=1'
c                 print *,'i j element=',i,j
                    print *,'area i=',propsect(iarea(i))
                    print *,'area j=',propsect(iarea(j))
                    print *,'Element SED below tolerance'
c                   pause
                   Else !above tolerance so skip, still need more iterations
                    print *,'i j element=',i,j
                    print *,' !above tolerance so skip, still need more iterations'
c                   pause
                    Goto 35
                   Endif
370                Continue
360               Continue

                  Do 91 i=1,inumel
                   print *,'chk do 91'
                   print *,'exlk=',exlk
c                  pause
                                  !If(i .eq. inumel .and. exlk .gt. .99)then !last element to check!
                                  print *,'SED below tolerance, OK, Stop!'
                     print *,'Reached optimial design'
                     pause
                     Stop
                     Endif
91                Continue
c                 goto 35

cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
c..If this is zero then some of the SED tolerance betweeen elements is
c..to big, so do not ratio the areas...
c  If no zeros, than lets ratio the areas.
cx                If(iSEDchk(i) .eq. 0)goto 38
cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                  If(exlk .lt. 1.0) then
300                 Do 311 i=1,nsect
c                   print *,'i=',i
C... Below is the updated Area
c                   print *,'propsect before/exlk=',propsect(i)
                    propsect(i)=propsect(i)/exlk
                    print *,'propsect after/exlk=',propsect(i)
                    pause
c...Amin = min area, do not allow areas to go below this minimum!
                     If(propsect(i) .lt. Amin)then
                     propsect(i)=Amin
                                  minarea(i)=1
                  print *,'minarea=1 for sect=',i
                  pause
                     Endif
311                 Continue
                   goto 38 !go to the end
                   Endif

35                Continue !Dummy statement

c.....
                  print *,'iexlk=',iexlk
                  If(iexlk .eq. 0) SSP=.1              !This means the force inc has gone to 1.0
                  If(iexlk .eq. 1) SSP=exlkdl0
                  If(iexlk .eq. 2) SSP=exlkdli
                  If(iexlk .eq. 3) SSP=exlkdlk
                    print *,'exlkdl0=',exlkdl0
                    print *,'exlkdli=',exlkdli
                    print *,'exlkdlk=',exlkdlk
                    print *,'SSP=',SSP
c.....

                  Do 33 j=1,inumel
                   SEDtotal =SEDtotal + strainD(j) !strain energy density
                   SEDtotal2=SEDtotal2 + strainD(j)**2. !sum of strain energy density squared
33    Continue
                  print *,'iarea(i)=',iarea
c__iarea(i) is an array listing the section prop number for each member
c__the prop's are listed according to the order of the members
                  Do 31 i=1,inumel
                  iprop=iarea(i) !number of section properties
                  print *,'iprop=',iprop
c                 pause
```

```
                    Do 32 j=1,i
c                    print *,'j=',j
c                   print *,'kprop(j)=',kprop(j)
                     If(kprop(j).eq.iprop)Then
c                    print *, 'goto 31' ! If we already updated the area, skip
c                    pause
                     goto 31 ! If we already updated the area, skip
                     Endif
32    Continue
C... Below is the updated Area
                    SEDscale=((SEDtotal/SEDtotal2)*SED(i))**SSP
                    print *,'SEDscale=',SEDscale
                    print *,'propsect before(_)=',propsect(iprop)
                    propsect(iprop)=propsect(iprop)*SEDscale
                    kprop(i)=iprop
                    print *,'propsect after(_)=',propsect(iprop)
c                   pause
c...Amin = min area, do not allow areas to go below this minimum!
                     If(propsect(iprop) .lt. Amin)then
                     propsect(iprop)=Amin
                                  minarea(iprop)=1
                      print *,'A minarea=1 for section property=',iprop
                   print *,'propsect(_)=',propsect(iprop)
c                     pause
                     Endif
31                  Continue
c.....................................................
38   Continue !dummy

                   print *,'propsect(_) at end=',propsect
                   print *,'iSEDchk(_) at end=',iSEDchk
                   Do 39 i=1,inumel
39                 print *,'SED(i) at end=',i,SED(i)

c                  pause

     deallocate (ie,je,iboundc,iet,
    $    jet,ip,iu,iut,iup,isupn,lm,iperm,ia)
                     deallocate (ja)
     deallocate ( b,be,ac,ad,an,dqtot,
    $    x,y,z,propmat,tempo1,tempo2,di,exfor)
     deallocate (disp,qk)
     deallocate (strainel,strainE,strainD,weight)
                     deallocate (Anew,Anew2,Anew3,
    $         Apart,SEDsum,Fin1,Fin2)

                   N=N+1 !This counts how many times thru rikarc!
                   print *,'N times thru rikarc=',N
                   iopt=1 !if =1 then going thru rik again for optimizer!
                    icxlk=0 !reset to zero
                   goto 1000

     stop
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
     subroutine elmass(rho,area,xl,lump,elm,ndofpe,maxdofpe,
    $cx,cy)
     implicit real*8(a-h,o-z)
c......get the 2-d truss element mass matrices
     dimension elm(maxdofpe,1)
     const=rho*area*xl
     do 1 i=1,ndofpe
      do 2 j=1,ndofpe
      elm(i,j)=0.
 2    continue
 1   continue
c
     if (lump .eq. 1) then    ! lumped mass is used
     do 3 i=1,ndofpe
     elm(i,i)=const/2.0
 3   continue
     elseif (lump .eq. 0) then    ! consistent mass is used
c
c    [m] = [lamdat] * [m_local] * [lamda]  -->> done by MATLAB
c    where:
c    [lamda] = [ cx  cy   0   0       ----->> Gere & Weaver's book
c             -cy  cx   0   0            page 255
c               0   0  cx  cy
c               0   0 -cy  cx ]
c    [m_local] = (rho*area*xl/6) * [ 2   0   1   0    -->> J.N. Reddy's book
c                                  0   0   0   0        page 233
c                                  1   0   2   0
c                                  0   0   0   0 ]
     const=const/6.0
     cxcx=cx*cx
     cycy=cy*cy
     cxcy=cx*cy
c
     elm(1,1)=const*cxcx*2.0
     elm(1,2)=const*cxcy*2.0
     elm(1,3)= elm(1,1)/2.0
     elm(1,4)= elm(1,2)/2.0
c
     elm(2,2)=const*cycy*2.0
     elm(2,3)=elm(1,4)
     elm(2,4)= elm(2,2)/2.0
c
     elm(3,3)=elm(1,1)
     elm(3,4)=elm(1,2)
c
     elm(4,4)=elm(2,2)
c
     do 12 i=1,ndofpe
      do 13 j=i,ndofpe
      elm(j,i)=elm(i,j)
 13   continue
 12  continue
c
     endif
      write(6,*) 'inside elmass: elm(i,i)= ',(elm(i,i),i=1,ndofpe)
     return
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     subroutine elstif(ndofpe,esm,be,propsect,iarea,
    $         area,youngm,xlength,cx,cy,cz,maxdofpe,
    $         xl,xli,deltaxi,deltayi,deltazi,x,y,z,
    $         iarc,xi,xj,yi,yj,zi,zj,ityp,lm,istep,iel,icheck,
    $         strainel,strainE,strainD,weight,Anew,Apart,Fin1,
    $         VED,VEDold,xlie,SED)

c     call elstif(ndofpe,elk,be,propsect,iarea,
c   $          area,youngm,xlength,cx,cy,cz,maxdofpe,
c   $          xl,xli,deltaxi,deltayi,deltazi,xx,yy,zz,
c   $          iarc,xi,xj,yi,yj,zi,zj,ityp,lm,istep,iel,icheck,
c   $          strainel,strainE,strainD,weight,Anew,Apart,Fin1,
c   $          VED,VEDold)

     implicit real*8(a-h,o-z)

c     common/elmatx/esm(6,6),cf(6),x(6),y(6),z(6),kl,ndofpe
     common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
     common/disp/ux(2),uy(2),uz(2),P(6),B1(6)
c     common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)

c......get the 2-d truss element stiffness, and element load vector !
     dimension be(*),esm(6,6),lm(1)
c      dimension be(*),esm(maxdofpe,*),lm(1)

                    dimension C(6),propsect(*),iarea(*)
c                   dimension P(6), B1(6)

c                   dimension x(3),y(3),z(3)
                    dimension x(*),y(*),z(*)
                   dimension strainel(*),strainE(*),strainD(*),weight(*),SED(*)
                   dimension Anew(*),Apart(*),Fin1(*),VED(*),VEDold(*),xlie(*)
c                  print *,'iel at start elstif=',iel !this is the element number
c                  print *,'xli at start elstif=',xli !total length of element updated
c                  print *,'lm1 at start elstif=',(lm(ic),ic=1,6)
c                  print *,'Start of Elstif'
c                  pause

                   icheck=0 !gah use to chk big stiff matrix when iel=4 with 2" axial disp.

     If(ityp .eq. 12)then
     DO 22 I=1,6
     DO 22 J=1,6
   22 esm(I,J)=0.0
     Do 11 I=1,2
     finx(i)=0.0
     finy(i)=0.0
   11 finz(i)=0.0
     endif

c  ityp = 6 for 2-D, =12 for 3-D
c     ityp = 12
c     itruss = 1 for Green's,  = 2 for Engineering
                   itruss = 1
c                  itruss = 2
c                  itruss = 3
                   POISS = .5d0

c                   print *,'iarc inside elstif=',iarc
c                  print *,'area inside elstif=',area
c                  print *,'youngm inside elstif=',youngm
c                  print *,'maxdofpe=',maxdofpe
c                  stop
c                  print *,'ityp inside elstif=',ityp

c                  stop
                    If(iarc .eq. 1 .or. iarc .eq. 0)then
c***************************************************************
cxxx      If(ityp.eq.12)then  ! Space truss element

c     print *,'Reading in Space Truss Stiffness In elstif'

c...... Three Dimensional Truss Element Stiffness matrix
c       a=dxe
c       e=dye

c       a=area
c                  print *,'iel=',iel
c                  print *,'propsect(iel)=',propsect(iel)
c                  pause

c                    a=propsect(iel)
                     iprop=iarea(iel)
c                  print *,'iel=',iel
c                  print *,'iprop=',iprop
                     a=propsect(iprop)
                     e=youngm

c     print *,'a = ',a
c     print *,'e = ',e

c     pause
c     print *,'xi  xj  yi  yj  zi  zj= ',xi,xj,yi,yj,zi,zj

c                  pause

cc     print *,'ux(1) & ux(2) = ',ux(1),ux(2)
cc     print *,'uy(1) & uy(2) = ',uy(1),uy(2)
cc     print *,'uz(1) & uz(2) = ',uz(1),uz(2)
cx                  x(1)=xi
cx                  x(2)=xj
cx                  y(1)=yi
cx                  y(2)=yj
cx                  z(1)=zi
cx                  z(2)=zj
cx     print *,'x(1)  x(2) = ',x(1),x(2)
cx     print *,'y(1)  y(2) = ',y(1),y(2)
cx     print *,'z(1)  z(2) = ',z(1),z(2)
cx                 pause
cx                   deltax=x(2)-x(1)
```

```
cx       deltay=y(2)-y(1)
cx       deltaz=z(2)-z(1)
                    deltax=xj-xi
         deltay=yj-yi
         deltaz=zj-zi

c                   print *,'deltay=',deltay
c                   pause

c        deltax=-x(2)+x(1)
c        deltay=-y(2)+y(1)
c        deltaz=-z(2)+z(1)

c         deltaxi=x(2)-ux(2) - ( x(1)-ux(1) )
c         deltaxi=x(2)+ux(2) - ( x(1)+ux(1) )
cx          deltaxi=x(2)-x(1) + ux(2)-ux(1)
          deltaxi=xj-xi + ux(2)-ux(1)
cc      print *,'A check'
c         deltayi=y(2)-uy(2) - ( y(1)-uy(1) )
c         deltayi=y(2)+uy(2) - ( y(1)+uy(1) )
cx          deltayi=y(2)-y(1) + uy(2)-uy(1)
          deltayi=yj-yi + uy(2)-uy(1)
c         print *,'deltayi=',deltayi
cc          print *,'y(2) y(1) = ',y(2),' ',y(1)
cc          print *,'uy(2) uy(1) = ',uy(2),' ',uy(1)

c         deltazi=z(2)-uz(2) - ( z(1)-uz(1) )
c         deltazi=z(2)+uz(2) - ( z(1)+uz(1) )
cx          deltazi=z(2)-z(1) + uz(2)-uz(1)
          deltazi=zj-zi + uz(2)-uz(1)
c         print *,'deltax=',deltax
c         print *,'deltay=',deltay
c         print *,'deltaz=',deltaz
c         print *,'deltaxi=',deltaxi
c         print *,'deltayi=',deltayi
c         print *,'deltazi=',deltazi
c                    pause
c       print *,'B check'
c                    pausc
         xl=dsqrt(deltax**2.d0 + deltay**2.d0 + deltaz**2.d0)
         xli=dsqrt(deltaxi**2.d0 + deltayi**2.d0 + deltazi**2.d0)
c       print *,'xli inside elstif=',xli
c                    pause

                     xlie(iel)=xli
c                   print *,'xlie(iel)=',xlie(iel)
                     density=.1d0
                     weight(iel)=density*A*xl
c                                print *,'icl inside elstif=',iel
c        print *,'xl inside elstif=',xl
c                    print *,'weight(iel)=',weight(iel)
c                    pause

c...Strain in element
c                    strainel(iel)=(xli-xl)/xl
                     strainel(iel)=(xli**2.-xl**2.)/(2.d0*xl)

c...Strain Energy
c                    strainE(iel)=.5*A*E*((strainel(iel))**2.d0)*xl
cc.. Strain Energy=(1/2)EAL(strain**2)
                     strainE(iel)=.5*((A*E)/xl)*(xli-xl)**2.

c...Strain Energy Density of ith element
cc                   strainD(icl)=strainE(iel)/(density*A*xl)
ccc                  strainD(icl)=strainE(iel)/(density)
ccc                  strainD(iel)=(strainE(iel)/(A*xl))/density

                     strainD(icl)=strainE(iel)/weight(iel)

                     SED(iel)=(.5*E/density) * ((xli-xl)/xl)**2.0
c                   print *,'iel inside elstif=',iel
c                   print *,'SED(iel) inside elstif=',SED(iel)
c                   pausc
c                    Anew(iel)=A*(1+(xli-xl)/xl)
c     Anew(icl)=A
                    Apart(iel)=((xli-xl)/xl)

c                   print *,'strainel(iel)=',strainel(iel)
c                   print *,'strainE(iel)=',strainE(iel)
c                   print *,'strainD(iel)=',strainD(iel)

c         deltaxi2=x(2)-ux(2) - ( x(1)-ux(1) )
c         deltayi2=y(2)-uy(2) - ( y(1)-uy(1) )
c         deltazi2=z(2)-uz(2) - ( z(1)-uz(1) )
c        xli2=dsqrt(deltaxi2**2.d0 + deltayi2**2.d0 + deltazi2**2.d0)
cc                  print *,'iel = ',iel
c        print *,'xl = ',xl
c        print *,'xli = ',xli
c                   stop
c        print *,'xli2 = ',xli2

c       print *,'x =',(x(ic),ic=1,4)
c       print *,'y =',(y(ic),ic=1,4)
c       print *,'z =',(z(ic),ic=1,4)
c                   pause

c                   print *,'lm2 in elstif=',(lm(ic),ic=1,6)
c                   pause


c                    stop

c      print *,'xl = ',xl
c      print *,'xli = ',xli
c      print *,'deltax = ',deltax
c      print *,'deltay = ',deltay
c      print *,'deltaz = ',deltaz
c      print *,'deltaxi = ',deltaxi
c      print *,'deltayi = ',deltayi
c      print *,'deltazi = ',deltazi
c      pausc
cc      print *,'check el 1'

c c      pause
c        stop
         cx=deltax/xl
         cy=deltay/xl
         cz=deltaz/xl
         const=e*a/xl
c      print *,'cx cy e a const',cx,cy,e,a,const
c        stop
       AN = 0.0
       ALAM = xl/xli
       X21D = deltaxi
       Y21D = deltayi
       Z21D = deltazi
c      print *,'X21D=',X21D
c      print *,'Y21D=',Y21D
c      print *,'Z21D=',Z21D

c                   print *,'lm2a in elstif=',(lm(ic),ic=1,6)
c                   pause

c      pause
c      If(ityp .eq. 12) then
       C(1) = -X21D
       C(2) = -Y21D
       C(3) = -Z21D
       C(4) = -C(1)
       C(5) = -C(2)
       C(6) = -C(3)
c                   print *,'C(1)=',C(1)
c                   print *,'C(2)=',C(2)
c                   print *,'C(3)=',C(3)
c                   print *,'C(4)=',C(4)
c                   print *,'C(5)=',C(5)
c                   print *,'C(6)=',C(6)

c                   print *,'lm2b in elstif=',(lm(ic),ic=1,6)
c                   pause

c                   pause
c      C(1) = X21D
c      C(2) = Y21D
c      C(3) = Z21D
c      C(4) = -C(1)
c      C(5) = -C(2)
c      C(6) = -C(3)

c**********************************
c      P(1)=-ux(1)
c      P(2)=-uy(1)
c      P(3)=-uz(1)
c      P(4)=-ux(2)
c      P(5)=-uy(2)
c      P(6)=-uz(2)
       P(1)=ux(1)
       P(2)=uy(1)
       P(3)=uz(1)
       P(4)=ux(2)
       P(5)=uy(2)
       P(6)=uz(2)
       dx= ux(2)-ux(1)
       dy= uy(2)-uy(1)
       dz= uz(2)-uz(1)

c                   print *,'lm2c in elstif=',(lm(ic),ic=1,6)
c                   pause

c      print *,'dx=',dx
c      print *,'dy=',dy
c      print *,'dz=',dz
       ALO2=xl*xl
c      B1(1) = X21D/ALO2
c      B1(2) = Y21D/ALO2
c      B1(3) = Z21D/ALO2
c      B1(4) = B1(1)
c      B1(5) = B1(2)
c      B1(6) = B1(3)
C..Glenn's modify
       B1(1) = -deltax/ALO2
       B1(2) = -deltay/ALO2
       B1(3) = -deltaz/ALO2
       B1(4) = -B1(1)
       B1(5) = -B1(2)
       B1(6) = -B1(3)

c                   print *,'lm2d in elstif=',(lm(ic),ic=1,6)
c                   pause

c      print *,'B1(_)=',B1
c      print *,'P(_)=',P
c                   pause
c                   stop

       EGR=0.D0
       Do 100 I=1,6
 100   EGR=EGR+B1(I)*P(I)
c      print *,'EGR1 = ',egr
c      EGR = B1(1)*P(1)+B1(2)*P(2)+B1(3)*P(3)+B1(4)*P(4)
c     c      +B1(5)*P(5)+B1(6)*P(6)
c      print *,'EGR = ',egr
c      pausc
c        print *,'dx=',dx
c        print *,'dy=',dy
c        print *,'dz=',dz

c                   print *,'lm3 in elstif=',(lm(ic),ic=1,6)
c                   pause

c       pausc
       EGR=EGR+.5*(dx*dx+dy*dy+dz*dz)/ALO2

       EST=EGR
c                   print *,'EST=',est
```

```
c                   pause
c                   stop
      If (itruss .eq. 2)then
                    Print *,'Using Engineering Strain Element!'
c                   STOP
c_____________________________________
      EST=2.*xl*EGR/(xli+xl)
c_____________________________________
      Endif

      If (itruss .eq. 3)then
                    Print *,'Using Log-Strain Element!'
c                   STOP
c_____________________________________
      EST=LOG(xli/xl)
                    print *,'EST=',est
c                   pause
c_____________________________________
      Endif


cc     print *,'EST = ',est
c** "AN" is the internal force from Crisfield, vol I, p.88
      AN = E*A*EST

                    Fin1(iel)=AN !This puts element Internal Force in array "Fin"

c                   If(N .eq.1 .or. N .eq.3 .or. N .eq.5 .or. N .eq.7)then
                    VED(iel)=fin1(iel)/(E*A**2.)
c                   print *,'VED(iel)=',VED(iel)
c                   Endif

c                   print *,'E=',E
c                   print *,'A=',A
c     print *,'AN = ',AN
c      print *,'xl = ',xl
c      print *,'xli = ',xli
c                   pause
c                    stop

c     If(Ifirst .eq. 1)then
c     write(12,*)'Internal Force AN'
c     endif
c     write(12,51) AN
c 51  format(e12.4)
c     print *,'AN2 = ',AN2
c**************************************
c     Endif

      EA = E*A
c      AN2 = A*E*(xl-xli)/xl

c                   stop
c     print *,'AN2 = ',AN2
c     pause
cc      CON1 = 1.0D0/xlorig**3
      CON1 = 1.0D0/xl**3
c** EA is very large compared to AN*ALAM
c                   print *,'itruss inside elstif = ',itruss

c                   print *,'lm3a in elstif=',(lm(ic),ic=1,6)
c                   pause

c                   print *,'itruss=',itruss

      If(itruss .eq. 1)then
      CON1 = E*A*CON1
c                   print *,'E=',E
c                   print *,'A=',A
c                   print *,'CON1 = ',CON1
c                   pause
      elseif(itruss .eq. 2)then
c                   ALAM=xl/AN
      CON1 = ALAM*ALAM*CON1*(EA-AN*ALAM)
      endif

                    If(itruss .eq. 3) then
                    CON1=CON1*ALAM**4*(EA-(1.d0+.5d0*POISS)*AN)
                    endif

c                   print *,'lm3b in elstif=',(lm(ic),ic=1,6)
c                   pause

c     print *,'CON1 & CON2 = ',con1,'  ',con2
cc     pause
      DO 33 J=1,6
      DO 33 I=1,J
c                   print *,'I=',i,' J=',j
c     print *,'C(I) & C(J) & CON1 = ',c(i),c(j),con1

c                   print *,'lm3bc in elstif=',lm(J)
c                   pause

   33 esm(I,J)=CON1*C(I)*C(J)

c                   print *,'lm3c in elstif=',(lm(icc),icc=1,6)
c                   pause

cvv      If(itruss .ge. 1)then
      If(itruss .eq. 1)then
      CON2=AN/xl
c                   print *,'CON2 =',con2
c                   stop
cvv      Elseif(itruss .eq. 2)then
      Else
      CON2=CON2*ALAM
      Endif
c                     print *,'CON2 =',con2
c                   pause
c      print *,'xl = ',xl
c      print *,'xli = ',xli

c                   print *,'lm3d in elstif=',(lm(ic),ic=1,6)
c                   pause

c                     stop

c     con2=0.
c..Crisfield original..
c     esm(1,1)=esm(1,1)+CON2
c     esm(2,2)=esm(2,2)+CON2
c     esm(3,3)=esm(3,3)+CON2
c     esm(4,4)=esm(4,4)+CON2
c     esm(5,5)=esm(5,5)+CON2
c     esm(6,6)=esm(6,6)+CON2
c     esm(1,4)=esm(1,4)-CON2
c     esm(2,5)=esm(2,5)-CON2
c     esm(3,6)=esm(3,6)-CON2

c..Duc's.....
      esm(1,1)=esm(1,1)+CON2
      esm(2,2)=esm(2,2)+CON2
      esm(3,3)=esm(3,3)+CON2
      esm(4,4)=esm(4,4)+CON2
      esm(5,5)=esm(5,5)+CON2
      esm(6,6)=esm(6,6)+CON2
      esm(1,4)=esm(1,4)-CON2
      esm(2,5)=esm(2,5)-CON2
      esm(3,6)=esm(3,6)-CON2


c                   If(istep .eq. 440)then
c                   print *,'CON2=',con2
c                   print *,'esm(1,1)=',esm(1,1)
c                   print *,'esm(2,2)=',esm(2,2)
c                   print *,'esm(3,3)=',esm(3,3)
c                   print *,'esm(4,4)=',esm(4,4)
c                   print *,'esm(5,5)=',esm(5,5)
c                   print *,'esm(6,6)=',esm(6,6)
c                   print *,'esm(1,4)=',esm(1,4)
c                   print *,'esm(2,5)=',esm(2,5)
c                   print *,'esm(3,6)=',esm(3,6)
c                   pause
c                   endif
c                     print *,'xl = ',xl
c      print *,'xli = ',xli
c     stop
c     esm(1,2)=0.
c     esm(1,3)=0.
c     esm(1,5)=0.
c     esm(1,6)=0.
c     esm(2,3)=0.
c     esm(2,4)=0.
c     esm(2,6)=0.
c     esm(3,4)=0.
c     esm(3,5)=0.

c..Crisfield modified
c     esm(1,1)=esm(1,1)+CON2
c     esm(2,2)=esm(2,2)+CON2
c     esm(3,3)=esm(3,3)+CON2
c     esm(4,4)=esm(4,4)+CON2
c     esm(5,5)=esm(5,5)+CON2
c     esm(6,6)=esm(6,6)+CON2
c     esm(1,2)=esm(1,2)-CON2
c     esm(1,3)=esm(1,3)-CON2
c     esm(1,4)=esm(1,4)-CON2
c     esm(1,5)=esm(1,5)-CON2
c     esm(1,6)=esm(1,6)-CON2
c     esm(3,4)=esm(3,4)-CON2
c     esm(3,5)=esm(3,5)-CON2
c     esm(3,6)=esm(3,6)-CON2
c     esm(4,5)=esm(4,5)-CON2
c     esm(4,6)=esm(4,6)-CON2
c     esm(5,6)=esm(5,6)-CON2
c..Glenn's esm..
c     csm(1,1)=csm(1,1)+CON2
c     esm(1,2)=esm(1,2)-CON2
c     esm(2,2)=esm(2,2)+CON2
c     esm(3,3)=esm(3,3)+CON2
c     esm(3,4)=esm(3,4)-CON2
c     esm(4,4)=esm(4,4)+CON2
c     esm(4,5)=esm(4,5)-CON2
c     esm(5,5)=esm(5,5)+CON2
c     esm(5,6)=esm(5,6)-CON2
c     esm(6,6)=esm(6,6)+CON2
c       print *,'xl = ',xl
c       print *,'xli = ',xli

      DO 344 I=1,6
      DO 344 J=1,6
c                   print *,'I',i,'J',j,'esm(I,J) inside elstif=',esm(i,j)
344   esm(J,I)=esm(I,J)

c                   print *,'lm4 in elstif=',(lm(ic),ic=1,6)
c                   pause

c                   pause

c       print *,'xl = ',xl
c       print *,'xli = ',xli
c       stop

c34     print *,'esm inside elstmf= ',i,' ',j,' ',esm(i,j)
c     pause

c     print *,'xli inside elstmf = ',xli
c     print *,'deltaxi inside elstmf = ',deltaxi
c      print *,'a & c inside elstmf=',a,e
c      print *,'xl inside elstmf=',xl

C                   print *,'itruss=',itruss
C                   PAUSE
c                    stop
```

```
c       print *,'xl = ',xl
c       print *,'xli = ',xli
c                       print *,'AN=',AN
c                       print *,'X21D=',x21d
c      stop

      If(itruss .eq. 1)Then
      Finx(1) = -AN * (X21D/xl)
      Finx(2) = AN * (X21D/xl)
      Finy(1) = -AN * (Y21D/xl)
      Finy(2) = AN * (Y21D/xl)
      FinZ(1) = -AN * (Z21D/xl)
      FinZ(2) = AN * (Z21D/xl)


cvv     Elseif(itruss .gt. 1)Then
cvv     Elseif(itruss .eq. 2)Then
      Else
      Finx(1) = -AN * (X21D/xli) * ALAM
      Finx(2) = AN * (X21D/xli) * ALAM
      Finy(1) = -AN * (Y21D/xli) * ALAM
      Finy(2) = AN * (Y21D/xli) * ALAM
      FinZ(1) = -AN * (Z21D/xli) * ALAM
      FinZ(2) = AN * (Z21D/xli) * ALAM
      Endif
c...deltaxi=X21D...AN = E*A*EST...
c      Finx(1) = -((xl-xli)*a*e/xl) * (deltaxi/xli)
c      Finx(2) = ((xl-xli)*a*e/xl) * (deltaxi/xli)
cc      Finx(1) = ((xl-xli)*a*e/xl) * (deltaxi/xli)
cc      Finx(2) = -((xl-xli)*a*c/xl) * (deltaxi/xli)
cc     print *,'AN axial load = ',AN
c     print *,'Finx(1) inside clstmf = ',finx(1)
c     print *,'Finx(2) inside elstmf = ',finx(2)
c     print *,'Finy(1) inside elstmf = ',finy(1)
c     print *,'Finy(2) inside elstmf = ',finy(2)
c     print *,'Finz(1) inside elstmf = ',finz(1)
c     print *,'Finz(2) inside elstmf = ',finz(2)
c     pause

c      Finy(1) = -AN * (Y21D/xl)
c      Finy(2) = AN * (Y21D/xl)
c      Finy(1) = -AN * (Y21D/xli)
c      Finy(2) = AN * (Y21D/xli)
cc     print *,'Finy(1) = -AN * (Y21D/xl)',finy(1)
cc     print *,'Finy(2) = AN * (Y21D/xl)',finy(2)

c      Finy(1) = -((xl-xli)*a*e/xl) * (deltayi/xli)
c      Finy(2) = ((xl-xli)*a*e/xl) * (deltayi/xli)
cc      Finy(1) = ((xl-xli)*a*e/xl) * (deltayi/xli)
cc      Finy(2) = -((xl-xli)*a*e/xl) * (deltayi/xli)

cc     print *,'xl = ',xl
cc     print *,'xli = ',xli
cc     print *,'deltayi = ',deltayi


cc     print *,'Finy(1) inside elstmf = ',finy(1)
cc     print *,'Finy(2) inside elstmf = ',finy(2)
c     pause

c      FinZ(1) = -AN * (Z21D/xl)
c      FinZ(2) = AN * (Z21D/xl)
c      FinZ(1) = -AN * (Z21D/xli)
c      FinZ(2) = AN * (Z21D/xli)
c      FinZ(1) = -((xl-xli)*a*e/xl) * (deltaZi/xli)
c      FinZ(2) = ((xl-xli)*a*e/xl) * (deltaZi/xli)
cc      FinZ(1) = ((xl-xli)*a*e/xl) * (deltaZi/xli)
cc      FinZ(2) = -((xl-xli)*a*e/xl) * (deltaZi/xli)

cc     print *,'FinZ(1) inside elstmf = ',finZ(1)
cc     print *,'FinZ(2) inside elstmf = ',finZ(2)
cc     pause

c     print *,'esm = ',esm
c     print *,'Finx & Finy & Finz inside elstmf = ',Finx,Finy,Finz
cc     print *,'deltayi = ',deltayi
cc     print *,'springK = ',springK
c     print *,'Inside elsmf at end'
c     pause
c       stop
      endif

c                       print *,'lm5 at end elstif=',(lm(ic),ic=1,6)
c                       pause

cxxx    Endif
c-------------------------------------------------------------
c       K=126
c       J=0
c       Do 101 I=1,126
c       K=J+K
c       J=126-I
c101    Continue
c       print *,'K=',K
c       stop

chk                     If(iel .eq. 4 .and. xli .le. 13.d0)then

                        icheck=1

chk                     print *,'iel =',iel
chk                     print *,'xli =',xli

c                       print *,'esm(1,1)=',esm(1,1)
c                       print *,'esm(2,2)=',esm(2,2)
c                       print *,'esm(3,3)=',esm(3,3)
c                       print *,'esm(4,4)=',esm(4,4)
c                       print *,'esm(5,5)=',esm(5,5)
c                       print *,'esm(6,6)=',esm(6,6)
c                       print *,'esm(1,4)=',esm(1,4)
c                       print *,'esm(2,5)=',esm(2,5)
c                       print *,'esm(3,6)=',esm(3,6)
c                       pause
chk                     endif
c                       pause

c************
c
c  output of the element matrices
c
c************
c************
  7   if(iplvl.eq.0) return
cgah     write(io,8) kk
cgah  8   format(/10x,7helement,i4/10x,12hforce vector,10x,
cgah    +16hstiffness matrix)
cgah     do 9 i=1,ndofpe
cgah  9   write(io,10) ef(i),(esm(i,j),j=1,ndofpe)
c  9   write(io,10) ef(i),(esm(i,j),j=1,kl)
cgah  10  format(8x,e8.2,3x,6e8.2)
      return
c*************
c*****************
c  diagnostic output
c******************
c******************
  5   continue


c  5   write(io,6) kk
c  6   format(//10x,19hthe area of element, i4,
c     +20h is less than 0.0001/
c     +10x,39hthe node numbers are in the wrong order/
c     +10x,33hor the nodes form a straight line/
c     +10x,20hexecution terminated)
      Ifirst=Ifirst+1 !counter for writing out internal forces
      stop
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
       subroutine info_elstif(iel,xlength,xl,cx,cy,cz,xx,yy,zz,ie,je
     $,propmat,propsect,youngm,area,rho,
     $ deltaxi,deltayi,deltazi,xli,
     $ xi,xj,yi,yj,zi,zj)

       implicit real*8(a-h,o-z)
       dimension xx(500),yy(500),zz(500),ie(1),je(1),
     $         propmat(1),propsect(1)
c       dimension ux(2),uy(2)

      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/disp/ux(2),uy(2),uz(2),P(6),B1(6)
c      common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)


c
c......purposes: obtain all necessary information before calculating
c            2-d truss element stiffness matrix (in global coordinate axis)
c
c                       print *,'Chk start info_elstif'
c                       print *,'xx start info_elstif=',(xx(i),i=1,4)
c                       print *,'yy start info_elstif=',(yy(i),i=1,4)
c                       print *,'zz start info_elstif=',(zz(i),i=1,4)
c                       print *,'ux(1)=',ux(1)
c                       print *,'uy(1)=',uy(1)
c                       print *,'uz(1)=',uz(1)
c                       print *,'ux(2)=',ux(2)
c                       print *,'uy(2)=',uy(2)
c                       print *,'uz(2)=',uz(2)
c                       pause
c                       stop
c                       pause
c                       print *,'iel =',iel
c                       print *,'ie =',ie
c                       print *,'je =',je
      locbegin=ie(iel)
c                       print *,'locbegin=',locbegin
c                       pause
c                       print *,'je(locbegin+1)=',je(locbegin+1)
c                       print *,'je(locbegin+3)=',je(locbegin+3)
c Below is for 2D truss
c                        nodei=je(locbegin+1)/2
c      nodej=je(locbegin+3)/2
c Changed for 3D truss
                         nodei=je(locbegin+2)/3
      nodej=je(locbegin+5)/3

c                       print *,'nodei & nodej=',nodei,'  ',nodej
c                       pause
      xi=xx(nodei)
c                       print *,'xi=',xi
c                       print *,'xx(nodei)=',xx(nodei)
c                       pause
c                       print *,'check after xi'
c                       pause
      yi=yy(nodei)
c                       print *,'yi=',yi
c                       print *,'check after yi'
c                       pause
                         zi=zz(nodei)
c                       print *,'zi=',zi
c                       print *,'check after zi'
c                       pause
      xj=xx(nodej)
c                       print *,'xj=',xj
c                       print *,'check after xj'
c                       pause
      yj=yy(nodej)
c                       print *,'yj=',yj
c                       print *,'check after yj'
c                       pause
                         zj=zz(nodej)
c                       print *,'zj=',zj
c                       print *,'check after zj'
```

```
c                 pause
c                 print *,'xi & yi & zi =',xi,' ',yi,' ',zi
c                 print *,'xj & yj & zj =',xj,' ',yj,' ',zj
c                 stop

      youngm=propmat(1)
      area=propsect(1)
      rho=propmat(10)

c*** Skip the rest of this, it is done in 'call elstif'***
c                 print *,'inside info_elstif...Skipping to end'
c                 pause
                                    GO TO 175
c*************************************************************
      deltax=xj-xi
      deltay=yj-yi
c                 print *,'deltay inside info_elstif=',deltay
c                 pause
                    deltaz=zj-zi
      xlength=dsqrt(deltax**2+deltay**2+deltaz**2)
c                 print *,'xlength inside info=',xlength
c                 stop

                  xl=xlength
      cx=deltax/xlength
      cy=deltay/xlength
                   cz=deltaz/xlength
c                 print *,'cx inside info_elstif=',cx
c                 print *,'cy inside info_elstif=',cy
c                 print *,'cz inside info_elstif=',cz
c                 pause
c
c                 print *,'inside info_elstif'
c                 print *,'ux(1)=',ux(1)
c                 print *,'uy(1)=',uy(1)
c                 print *,'uz(1)=',uz(1)
c                 print *,'ux(2)=',ux(2)
c                 print *,'uy(2)=',uy(2)
c                 print *,'uz(2)=',uz(2)
c                 pause

                   deltaxi=xj-ux(2)-(xi-ux(1))
                   deltayi=yj-uy(2)-(yi-uy(1))
cc                 deltayi=yj+uy(2)-(yi+uy(1))
                   deltazi=zj-uz(2)-(zi-uz(1))
                   xli=dsqrt(deltaxi**2.0+deltayi**2.0+deltazi**2.0)

c                 print *,'deltaxi inside info_elstif=',deltaxi
c                 print *,'deltayi inside info_elstif=',deltayi
c                 print *,'deltazi inside info_elstif=',deltazi
c                 print *,'xli=',xli
c                 pause

175   CONTINUE !dummy line
      return
      end
c=====================================================================
======
c     The following routines are extracted from file ducsparsesolver.f
c     see cd ~/cee/newfem/*completely*/
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine crnormd(answer,rhs,x,disamax,locate,disasum,n,relerr)
      implicit real*8(a-h,o-z)
      dimension answer(*),rhs(*),x(*)
c.....
      relerr=0.
      bnorm=0.
      disamax=0.
      disasum=0.
      do 1 i =1,n
      bnorm=bnorm+rhs(i)*rhs(i)
      diff=answer(i)-rhs(i)
      relerr=relerr+diff*diff
      absdis=dabs( x(i) )
      disasum=disasum+absdis
      if(absdis.gt.disamax) then
      disamax=absdis
      locate=i
      endif
 1    continue
      relerr=dsqrt(relerr)
      bnorm=dsqrt(bnorm)
      relerr=relerr/bnorm
                  print *,'inside crnormd'
                  pause
                  stop
      write(6,*) 'relerr,disamax,locate,disasum = '
      write(6,*) relerr,disamax,locate,disasum
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine fbod(n,iu,ju,di,un,b,x,iopb,isupd,getsolve)
      implicit real*8(a-h,o-z)
      dimension iu(*),ju(*),di(*),un(*),b(*),x(*),isupd(*)
C
c     input:    iu,ju,un    given upper triangular matrix with
c                   unit diagonal matrix U in RR(U)U.
c               di          inverses of the diagonal elements
c                           of the diagonal matrix D.
c               b           right-hand side vector b.
c               n           order of the system, N>1.
c     output:   x           vector of unknowns x.
c                 print *,'****** Start fbod ******'
c                 pause
c                 print *,'****** Start fbod ******'
c                 pause
c                 print *,'****** Start fbod ******'
c                 pause
c                 print *,'getstiff=',getstiff
c                 print *,'getsolve=',getsolve
c                 pause
c                 If(getsolve .eq. 1)Then
c     di(12)=1./544453.3715
c                 Endif

c                 print *,'N inside fbod=',N
c                 print *,'IU(_) = ',(iu(ii),ii=1,N-1)
c                 pausc
c                 print *,'JU(_) = ',(ju(ii),ii=1,N-1)
c     pause
c                 print *,'di start fbod=',(di(ii),ii=1,N)
c     pause
c                 print *,'UN(_) = ',(un(ii),ii=1,N-1)
c                 pause
c                 print *,'b(_)=',(b(ii),ii=1,N)
c                 pause

c                 write(7,*)'N inside fbod=',N
c                 write(7,*)'IU(_) = ',(iu(ii),ii=1,N-1)
c                 write(7,*)'JU(_) = ',(ju(ii),ii=1,3)
c                 write(7,*)'di=',(di(ii),ii=1,N)
c                 write(7,*)'UN(_) = ',(un(ii),ii=1,N-1)
c                 write(7,*)'b(_)=',(b(ii),ii=1,12)
c                 pause
      NM=N-1

      DO 10 I=1,N
c                 print *,'B(I) inside fbod=',B(I)
c                 write(7,*)'B(I) inside fbod=',B(I)
  10  X(I)=B(I)
c                 pause

      DO 40 K=1,NM
c                 print *,'K=',k
c                 print *,'NM=',nm
c                 write(7,*)'K=',k
c                 write(7,*)'NM=',nm
      IUA=IU(K)
      IUB=IU(K+1)-1
      XX=X(K)
c                 print *,'X(K)=',x(k)
c                 print *,'IUA=',iua
c                 print *,'IUB=',iub
c                 pause
c                 write(7,*)'X(K)=',x(k)
c                 write(7,*)'IUA=',iua
c                 write(7,*)'IUB=',iub

      IF(IUB.LT.IUA)GO TO 30

      DO 20 I=IUA,IUB
c                 print *,'I=',i
c                 pause
c                 print *,'JU(I)=',ju(i)
c                 write(7,*)'JU(I)=',ju(i)
c                 print *,'X(JU(I))=',X(JU(I))
c                 write(7,*)'X(JU(I))=',X(JU(I))
c                 print *,'UN(I)=',UN(I)
c                 write(7,*)'UN(I)=',UN(I)
20      X(JU(I))=X(JU(I))-UN(I)*XX

c                 print *,'X(JU(I))=',(x(ic),ic=1,n)

c                  print *,'JU(I)=',ju(i)
c                  print *,'X(JU(I))=',x(ju(i))
c                 write(7,*)'JU(I)=',ju(i)
c                 write(7,*)'X(JU(I))=',x(ju(i))
c                  pause
c                 print *,'XX=',xx
c                 write(7,*)'XX=',xx
c                 print *,'DI(K)=',DI(K)
c                 write (7,*)'DI(K)=',DI(K)
30      X(K)=XX*DI(K)
c     print *,'X(K)=',(X(ic),ic=1,n)

c                  print *,'XX=',xx
c                  print *,'DI(K)=',DI(K)
c                  print *,'X(K)=',x(K)
c                                  write(7,*)'XX=',xx
c                                  write(7,*)'DI(K)=',DI(K)
c                                  write(7,*)'X(K)=',x(K)
  40 CONTINUE
c                 pause

c                 print *,'X(5)=',x(5)
c                 print *,'X(N) before=',X(N)
      X(N)=X(N)*DI(N)
c                 print *,'DI(N)=',DI(N)
c                 print *,'N=',n
c                 print *,'X(N)=',x(n)
c                 write(7,*)'X(N)=',x(n)
c                 pause
      K=NM
  50  IUA=IU(K)
      IUB=IU(K+1)-1
      IF(IUB.LT.IUA)GO TO 70
c                 print *,'X(K)=',x(k)
c                 write(7,*)'X(K)=',x(k)
      XX=X(K)
c                 print *,'X(5)=',x(5)
      DO 60 I=IUA,IUB
c                  print *,'XX=',xx
c                  print *,'IUA=',iua
c                  print *,'IUB=',iub
c                 write(7,*)'XX=',xx
c                 write(7,*)'IUA=',iua
c                 write(7,*)'IUB=',iub
c                 pause
c                  print *,'XX=',xx
c                  print *,'UN(I)=',un(i)
c                  print *,'JU(I)=',ju(i)
c                  print *,'X(12)=',x(12)
```

```
c                    print *,'X(JU(I))=',x(ju(i))
c                    write(7,*)'XX=',xx
c                    write(7,*)'UN(I)=',un(i)
c                    write(7,*)'JU(I)=',ju(i)
c                    write(7,*)'X(12)=',x(12)
c                    write(7,*)'X(JU(I))=',x(ju(i))
c                    pause
  60 XX=XX-UN(I)*X(JU(I))
c                    print *,'XX=',xx
c                    print *,'K=',k
c                    write(7,*)'XX=',xx
c                    write(7,*)'K=',k
c                    pause
   X(K)=XX
  70 K=K-1
   IF(K.GT.0)GO TO 50
c     write(6,*) 'displacement solution= '
c     write(7,*) 'displacement solution= '
c     write(6,4) (x(i1),i1=1,n)
c                    write(*,4) (x(i2),i2=1,n)
c 4    format(5(e13.6,1x))
c                    print *,'x = ',(x(i3),i3=1,n)
c                    print *,'End of fbed'
c                                    write(7,*)'x = ',(x(i3),i3=1,n)
c                                    write(7,*)'End of fbed'
c                    stop
     return
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     subroutine mvspar(diag,     kindx,coefs,rhs,istartr       ! no need array kptrs(-)
    $,n,ncoef,answer,ioper,isupern)
c*************************************************************
c     PLEASE do "NOT" distribute this source code to any
c     persons/organizations, unless obtaining the written
c     permission from Duc T. Nguyen (Tel= 757-683-3761,
c                          Email= dnguyen@odu.edu)
c*************************************************************
     implicit real*8(a-h,o-z)
c     implicit complex*16(a-h,o-z)
c......purpose: <sparse, and symmetric> matrix times vector
c......       unrolling option will soon be provided
c......       This routine has been verified to give "CORRECT" answer (June 30'00)
c     dimension diag(1),kptrs(1),kindx(1),coefs(1),rhs(1),istartr(1)
     dimension diag(1),       kindx(1),coefs(1),rhs(1),istartr(1)! no need array kptrs(-)
    $,answer(1),isupern(1)
c
c......for consecutive rows,determine if ALL nonzero(s) off-diagonals
c......are lined up in the same columns ! (for "FUTURE" unrolling option !)
c     call supern(n,kptrs,istartr,kindx,isupern)
c......initialize
     ioper=0
c
     do 3 i=1,n
 3     answer(i)=diag(i)*rhs(i)
     ioper=ioper+n
c......begin SPARSE matrix times vector
     do 1 i=1,n
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
c     ii=kptrs(i)             ! no need to have kptrs(-) array
     ii=istartr(i+1)-istartr(i)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
     if (ii.eq.0) go to 1
     icount=istartr(i)-1
     do 2 k=1,ii
c.......row i-th (upper triangular) operates on vector rhs(-)
     icount=icount+1
c.......find the column numbers (of the i-th row) which have non-zero
c.......off-diagonals
     jcoln=kindx(icount)
     answer(i)=answer(i)+coefs(icount)*rhs(jcoln)
     ioper=ioper+2
c.......now, column i-th (lower triangular) operates on vector rhs(-)
     answer(jcoln)=answer(jcoln)+coefs(icount)*rhs(i)
     ioper=ioper+2
 2    continue
 1    continue
     return
     end


C*****************************************************************
     subroutine numfa1d(n,ia,ja,ad,an,iu,ju,di,un,ip,iup,isupd,iopf,
    $               istep)
c     subroutine numfa1d(n,ia2,ja,ad,an,iu,ju,di,un,ip,iup,isupd,iopf)
     implicit real*8(a-h,o-z)
c     implicit complex*16(a-h,o-z)
     dimension ia(*),ja(*),ad(*),an(*),iu(*),ju(*),di(*),un(*)
c     dimension ia2(*),ja(*),ad(*),an(*),iu(*),ju(*),di(*),un(*)
     dimension ip(*),iup(*),isupd(*)
C......purpose: numerical factorization
c   input:      ia,ja,an,ad    given matrix A in RR(U)U.
c             iu,ju         structure of resulting matrix U in
c                           RR(U)O.
c             n             order of matrices A and U.
c   output:     un          numerical values of the nonzeros of
c                           matrix U in RR(U)O.
c             di            inverse of the diagonal matrix D.
c   working space:  ip       of dimension N.  Chained lists of rows
c                           associated with each column.
c               iup         of dimension N.  Auxiliary pointers to
c                           portions of rows.
c               di          is used as the expanded accumulator.
c                    print *,'****** start of numfa1d *******'
c                    pause
c                    print *,'****** start of numfa1d *******'
c                    pause

c                    print *,'****** start of numfa1d *******'
c                    pause
c                    print *,'n start numfa1d=',n
c                    pause

     ncoef1=ia(n+1)-1

c                    If(istep .eq. 1)then
c                    print *,'ncoef1 inside numfa1d',ncoef1
c                    pause
c     endif

c     ncoef1=ia2(n+1)-1
c     print *, 'inside numfa1: ad(-) = ',(ad(ic),ic=1,n)
c                    pause
c     print *, 'inside numfa1: an(-) = ',(an(ic),ic=1,ncoef1)
c     print *, 'inside numfa1: an(-) = ',(an(ic),ic=1,30)
c                    pause

c                    print *,'ia2(_) inside numfa1=',(ia2(i),i=1,n)
c                    stop
c     write(6,*) 'inside numfa1: ad(-) = ',(ad(ic),ic=1,n)
c     write(6,*) 'inside numfa1: an(-) = ',(an(ic),ic=1,ncoef1)
c                    print *,'ip(_)=',(ip(ic),ic=1,9)
c                    print *,'JU(_)=',(ju(ic),ic=1,3)
c                    print *,'UN(_)@start numfa1d=',(un(ic),ic=1,n)
c                    print *,'DI(_)@start numfa1d=',(ju(ic),ic=1,n)
c                    pause
c                                    print *,'UN(1)=',un(1)
c                                    print *,'UN=',un
c     print *, 'inside numfa1: IU(-) = ',(IU(ic),ic=1,n)

c                    pause
c

     DO 10 J=1,N
  10 IP(J)=0
c......Begin of of 1-st (nested) loop: outer-most loop, for each i-th row
     DO 130 I=1,N
c                    print *,'I @ start 1st loop=',i
c                    pause
c     print *, 'outer-most loop 130 I= ',i
c     print *,'outer-most loop 130 I= ',i
     IH=I+1
     IUA=IU(I)
     IUB=IU(IH)-1
c                    print *,'IUA inside NUMFAC=',iua
c                    print *,'IUB inside NUMFAC=',iub
c                    pause
c     print *, 'NUMFAC: i,n,ih,iu(i),iua,iu(ih),iub ='
c     print *, i,n,ih,iu(i),iua,iu(ih),iub

c     write(6,*) 'NUMFAC: i,n,ih,iu(i),iua,iu(ih),iub ='
c     write(6,*) i,n,ih,iu(i),iua,iu(ih),iub
c                                    print *,'UN(1)before goto40=',un(1)
     IF(IUB.LT.IUA)GO TO 40

     DO 20 J=IUA,IUB
     DI(JU(J))=0.
c     write(6,*) 'NUMFAC,loop 20: j,iua,iub,ju(j),di(ju(j)) = '
c     write(6,*) j,iua,iub,ju(j),di(ju(j))
 20  continue
c                    print *,'di(ju(j))=',(di(ic),ic=ju(iua),ju(iub))
c                    pause

     IAA=IA(I)
     IAB=IA(IH)-1
c     IAA=IA2(I)
c     IAB=IA2(IH)-1

c     write(6,*) 'NUMFAC: i,ia2(i),iaa,ih,ia2(ih),iab= '
c     write(6,*) i,ia2(i),iaa,ih,ia2(ih),iab
     IF(IAB.LT.IAA)GO TO 40

     DO 30 J=IAA,IAB
c                                    print *,'UN(1)after 30=',un(1)
c 30 DI(JA(J))=AN(J)
c 40 DI(I)=AD(I)
     DI(JA(J))=AN(J)
c                    print *,'J inside numfa1d=',j
c                    pause
c                    print *,'AN(J) inside numfa1d=',AN(J)
c                    pause
c                    print *,'DI(JA(J)) 30 inside numfa1d=',DI(JA(J))
c                    pause
c     write(6,*) 'NUMFAC:loop30: j,iaa,iab,an(j),ja(j),di(ja(j))= '
c     write(6,*) j,iaa,iab,an(j),ja(j),di(ja(j))
 30   continue

 40   DI(I)=AD(I)
c                    print *,'DI(I) 40=',AD(I)
c                    pause
c                                    print *,'UN(1)after 40=',un(1)
c     write(6,*) 'NUMFAC: i,ad(i),di(i)= '
c     write(6,*) i,ad(i),di(i)
c                    print *,'NUMFAC: i,ad(i),di(i)=',i,ad(i),di(i)
     LAST=IP(I)
c                    print *,'LAST=',last
c                    pause

c     write(6,*) 'NUMFAC: i,ip(i),last= '
c     write(6,*) i,ip(i),last
c                                    print *,'UN(1)before goto90=',un(1)
     IF(LAST.EQ.0)GO TO 90

     LN=IP(LAST)
c                    print *,'LN=',LN

c     write(6,*) 'NUMFAC: last,ip(last),ln = '
c     write(6,*) last,ip(last),ln
c......begin of 2-nd (nested) loop: considering all APPROPRIATED previous
c......                     rows (any appropriated rows 1--->i-1)
  50 L=LN
     LN=IP(L)
```

```
      IUC=IUP(L)
      IUD=IU(L+1)-1
      UM=UN(IUC)*DI(L)
c                   print *,'IUC=',iuc
c                   print *,'IUD=',iud
c                   print *,'UM=',um

c     write(6,*) 'NUMFAC: l,ip(l),ln,iup(l),iuc = '
c     write(6,*) l,ip(l),ln,iup(l),iuc
c     write(6,*) 'NUMFAC: iu(L+1),iud,l,di(l),iuc,un(iuc),um= '
c     write(6,*) iu(L+1),iud,l,di(l),iuc,un(iuc),um
c......begin of 3-rd (nested) inner-most loop: considering all APPROPRIATED
c......                          columns (any columns i--->n)
c                   print *,'start 3rd nested loop'

c                   stop
c                   print *,'IUC=',iuc
c                   print *,'IUD=',iud
c                   print *,'UN(1)=',un(1)
      DO 60 J=IUC,IUD
c                   print *,'UN(1)after 60=',un(1)
      JJ=JU(J)
c     write(6,*) 'NUMFAC:loop60:J,di(jj)= '
c     write(6,*) J,di(jj)
c  60 DI(JJ)=DI(JJ)-UN(J)*UM
c                   print *,'DI(JJ)=',di(jj)
c                   print *,'J=',j
c                   print *,'UN(J)=',un(j)
c                   print *,'UM=',um
cccccccccccccccccccccccccccccccccccccccccccccccc
c                   UN(1)=0.0
cccccccccccccccccccccccccccccccccccccccccccccccc
c                   print *,'UN(1)a=',un(1)
      DI(JJ)=DI(JJ)-UN(J)*UM
c                   print *,'JJ=',jj
c                   print *,'DI(JJ)=',di(jj)

c                   write(6,*) 'NUMFAC:loop60:j,iuc,iud,ju(j),jj,um,un(j),di(jj)= '
c     write(6,*) j,iuc,iud,ju(j),jj,um,un(j),di(jj)
 60   continue
      UN(IUC)=UM
c                   print *,'UN(1)b=',un(1)
      IUP(L)=IUC+1
c                   print *,'UN(IUC)=',un(iuc)
c                   print *,'IUP(L)=',iup(l)

c     write(6,*) 'NUMFAC: um,iuc,iud,un(iuc),l,iup(l) = '
c     write(6,*) um,iuc,iud,un(iuc),l,iup(l)
      IF(IUC.EQ.IUD)GO TO 80
      J=JU(IUC+1)
      JJ=IP(J)
c     write(6,*) 'NUMFAC: iuc,ju(iuc+1),j,ip(j),jj= '
c     write(6,*) iuc,ju(iuc+1),j,ip(j),jj
      IF(JJ.EQ.0)GO TO 70
      IP(L)=IP(JJ)
c     write(6,*) 'NUMFAC: jj,ip(jj)= '
c     write(6,*) jj,ip(jj)
      IP(JJ)=L
c     write(6,*) 'NUMFAC: jj,ip(jj),l,ip(l)= '
c     write(6,*) jj,ip(jj),l,ip(l)
      GO TO 80
   70 IP(J)=L
      IP(L)=L
c     write(6,*) 'NUMFAC: l,j,ip(j),ip(l),last= '
c     write(6,*) l,j,ip(j),ip(l),last
c......the following go to statement is equivalent to 2-nd nested loop
c......for factorization
c......
c                   print *,'Check numfac 1'
c                   print *,'L=',L,'  LAST=',last
c                   stop
   80 IF(L.NE.LAST)GO TO 50
c......
 90   continue
c     write(6,*) 'di(i)= ', di(i)
c      print *, 'di(i)= ', di(i)
c                   pause

      DI(I)=1./DI(I)
c                    print *,'I=',i
c      print *, '1./di(i) after 90 = ', di(i)
c                   pause

c     write(6,*) 'NUMFAC: i,di(i)=1/di(i),iub,iua= '
c     write(6,*) i,di(i),iub,iua
c                   print *,'IUA=',iua
c                   print *,'IUB=',iub

ccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Changed Duc's code from " .LT. to .LE. "
      IF(IUB.LT.IUA)GO TO 120
c     IF(IUB.LE.IUA)GO TO 120
c     IF(IUB.GT.IUA)GO TO 120
ccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      DO 100 J=IUA,IUB
c 100 UN(J)=DI(JU(J))
c                                  print *,'UN(1)before chg=',un(1)
c                   print *,'JU(J)=',ju(j)
c                   print *,'DI(JU(J))=',di(ju(j))
      UN(J)=DI(JU(J))
c                                  print *,'UN(1)after chg=',un(1)
c                                  print *,'UN(1)c=',un(1)
c     write(6,*) 'NUMFAC:loop100: j,iua,iub,ju(j),di(ju(j)),un(j)= '
c     write(6,*) j,iua,iub,ju(j),di(ju(j)),un(j)
c                    print *,'I=',i
c      print *, '1./di(i) chk 1 = ', di(i)
c                   pause
 100  continue
      J=JU(IUA)
      JJ=IP(J)
c     write(6,*) 'NUMFAC: iua,ju(iua),j,ip(j),jj= '
c     write(6,*) iua,ju(iua),j,ip(j),jj
      IF(JJ.EQ.0)GO TO 110
      IP(I)=IP(JJ)
c     write(6,*) 'NUMFAC: jj,ip(jj)= '
c     write(6,*) jj,ip(jj)
      IP(JJ)=I
c     write(6,*) 'NUMFAC: jj,ip(jj),i,ip(i)= '
c     write(6,*) jj,ip(jj),i,ip(i)
      GO TO 120
  110 IP(J)=I
      IP(I)=I
c     write(6,*) 'NUMFAC: i,j,ip(j),ip(i)= '
c     write(6,*) i,j,ip(j),ip(i)
c                    print *,'I=',i
c      print *, '1./di(i) chk 2 = ', di(i)
c                   pause
  120 IUP(I)=IUA
c     write(6,*) 'NUMFAC: iua,i,iup(i)= '
c     write(6,*) iua,i,iup(i)
c                   print *,'NUMFAC: iua,i,iup(i)=',iua,i,iup(i)
c                   stop
C      print *,'check numfac 2'
c                    print *,'I=',i
c      print *, '1./di(i) chk 3 = ', di(i)
c                   pause
  130 CONTINUE

c     WRITE(6,*),'End of numfa1d'
c                   print *,'DI at end of numfa1d=',(di(iz),iz=1,n)
c                   print *,'End of numfa1d'
c                   print *,'**********************'
c                   pause
c                   print *,'**********************'
c                   pause
c                   print *,'**********************'
c                   pause
c                   print *,'**********************'
c                   pause
c                   stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
      subroutine symfactd(n,ia,ja,iu,ju,ip,ncoef2)
c      call symfactd(ndof,ia,ja,iu,itempo1,ip,ncoef2)
c      subroutine symfactd(n,ia2,ja,iu,ju,ip,ncoef2)
c     implicit real*8(a-h,o-z)
      dimension ia(*),ja(*),iu(*),ju(*),ip(*)
c      dimension ia2(*),ja(*),iu(*),ju(*),ip(*)
C......Purposes: Symbolic factorization
c     input:  ia(n+1),ja(ncoef1)   structure of given matrix A in RR(U)U.
c......      ia(n+1)           starting location of 1-st non-zero term
c            n                order of matrix A and of matrix U.
c......      ncoef1 = ia(n+1)-1
c     output: iu(n+1),ju(ncoef2)   structure of resulting matrix U in RR(U)U.
c     working space: ip(n)       of dimension N.  Chained lists of rows
c                         associated with each column.
c                         The array IU is also used as the multiple
c                         switch array.
c                   print *,'***************inside symfactd******************'
c     print *, 'inside symfactd: ia(-) = ',(ia(i),i=1,n+1)
c     print *, 'inside symfactd: ja(-) = ',(ja(i),i=1,n+1)
c     print *, 'inside symfactd: ja(-) = ',(ja(i),i=1,3)
c                   pause
c                    print *,'ia2 inside symfactd=',(ia2(i),i=1,n+1)
c                   stop

                    JI=0
                    JPP=0
                    MIN=0
                    IAA=0
                    IAB=0
                    JJ=0
                    JP=0
                    LAST=0
                    L=0
                    LH=0
                    IUA=0
                    IUB=0
                    NH=0
                    ncoef2=0
                    J=0
                    NM=0

      ncoef1=ia(n+1)-1
c      ncoef1=ia2(n+1)-1

c     print *, 'start symfactd: ia(-) = ',(ia(ic),ic=1,n+1)
c     print *, 'inside symfactd: ja(-) = ',(ja(i),i=1,ncoef1)
c     write(6,*) 'inside symfactd: ia2(-) = ',(ia2(i),i=1,n+1)
c     write(6,*) 'inside symfactd: ja(-) = ',(ja(i),i=1,ncoef1)
c                   pause
c
      NM=N-1
      NH=N+1
      DO 8 I=1,N+1
                    ju(i)=0
    8 ip(i)=0

      DO 10 I=1,N
      IU(I)=0
   10 IP(I)=0
      JP=1

      DO 90 I=1,NM
       JPI=JP
       JPP=N+JP-1
       MIN=NH
c                   print *,'I=',i
c                   print *,'JPI=',JPI
c                   print *,'JPP=',JPP
c                   print *,'MIN=',min
```

```
c                  pausc
c                   print *,'IA2(I)=',IA2(I)
c                   print *,'IA2(I+1)=',IA2(I+1)
c                  write(6,*)'I=',i
c                  write(6,*)'JPI=',JPI
c                  write(6,*)'JPP=',JPP
c                  write(6,*)'MIN=',min
c                  pause
c                   write(6,*)'IA2(I)=',IA2(I)
c                   write(6,*)'IA2(I+1)=',IA2(I+1)

c                   stop
      IAA=IA(I)
      IAB=IA(I+1)-1
c      IAA=IA2(I)
c      IAB=IA2(I+1)-1

c                  print *,'IAA=',iaa
c                  print *,'IAB=',iab
c                  write(6,*)'IAA=',iaa
c                  write(6,*)'IAB=',iab
c                  pause
c     write(6,*) 'SYMFACT: i,nm,jpi,jp,n,jpp,min,iaa,iab = '
c     write(6,*) i,nm,jpi,jp,n,jpp,min,iaa,iab
c      print *,'SYMFACT: i,nm,jpi,jp,n,jpp,min,iaa,iab = '
c      print *, i,nm,jpi,jp,n,jpp,min,iaa,iab

      IF(IAB.LT.IAA)GO TO 30
c                   print *,'check I inside symfacd'
c                  print *,'JA(_) inside symfacd=',(JA(ic),ic=1,3)
c                  print *,'IAA=',IAA
c                  print *,'IAB=',IAB
       DO 20 J=IAA,IAB
c                  print *,'JA(J)=',JA(J)
       JJ=JA(J)
c                  print *,'chk JA(J) inside symfacd'
       JU(JP)=JJ
c     print *,'chk JU(JP) inside symfacd'
c      write(6,*) 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c      write(6,*) j,jj,jp,ju(jp),min
c       print *, 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c       print *, j,jj,jp,ju(jp),min
c                  pausc
c                    stop

       JP=JP+1
c      write(6,*) 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c      write(6,*) j,jj,jp,ju(jp),min
c       print *, 'SYMFACT:insideloop20: j,jj,jp,ju(jp),min = '
c       print *, j,jj,jp,ju(jp),min

c      IF(JJ.LT.MIN)MIN=JJ
       IF(JJ.LT.MIN) then
       min=jj
c      write(6,*) 'SYMFACT:insideloop20: min = '
c      write(6,*) min
c       print *, 'SYMFACT:insideloop20: min = '
c       print *, min

       endif
c 20   IU(JJ)=I
c                   print *,'I=',i
c                   print *,'JJ=',JJ
       IU(JJ)=I
c      print *, 'SYMFACT:insideloop20: jj,iu(jj),i = '
c      print *, jj,iu(jj),i
c      write(6,*) 'SYMFACT:insideloop20: jj,iu(jj),i = '
c      write(6,*) jj,iu(jj),i
 20     continue

c                   print *,'chk 1a inside sysfactd'
c                  write(6,*)'chk 1a inside sysfactd'
c                   stop
  30  LAST=IP(I)
c     write(6,*) 'SYMFACT: i,ip(i),last = '
c     write(6,*) i,ip(i),last
      IF(LAST.EQ.0)GO TO 60
      L=LAST
c                  print *,'LAST=',last
c                  write(6,*)'LAST=',last
c                  pause
  40 L=IP(L)
c                  print *,'chk 1b inside sysfactd'
c                  print *,'L=',L
c                  print *,'IP(L)=',IP(L)
c                  print *,'IU(L)=',IU(L)
c                  write(6,*)'chk 1b inside sysfactd'
c                  write(6,*)'L=',L
c                  write(6,*)'IP(L)=',IP(L)
c                  write(6,*)'IU(L)=',IU(L)
c                  pause
      LH=L+1
      IUA=IU(L)
      IUB=IU(LH)-1
c                  print *,'IU(LH)=',IU(LH)
c                                        print *,'chk 1c inside sysfactd'
c                  write(6,*)'IU(LH)=',IU(LH)
c                  write(6,*)'chk 1c inside sysfactd'

c     write(6,*) 'SYMFACT: last,l,ip(l),lh,iu(l),iua,iu(lh),iub = '
c     write(6,*) last,l,ip(l),lh,iu(l),iua,iu(lh),iub
c     IF(LH.EQ.I)IUB=JPI-1
      IF(LH.EQ.I) then
c                                        print *,'chk 1d inside sysfactd'
      iub=jpi-1
c          write(6,*)'chk 1d inside sysfactd'
c     write(6,*) 'SYMFACT: lh,i,jpi,iub = '
c     write(6,*) lh,i,jpi,iub
       endif
c                  print *,'chk 1e inside sysfactd'
c                  write(6,*)'chk 1e inside sysfactd'
      IU(I)=I
c                  print *,'IU(I)=',IU(I)

c     write(6,*) 'SYMFACT: i,iu(i) = '
c     write(6,*) i,iu(i)
      DO 50 J=IUA,IUB
      JJ=JU(J)
c                  print *,'JJ=',jj
c     write(6,*) 'SYMFACT:insideloop50: j,iua,iub,ju(j),jj,iu(jj),i = '
c     write(6,*) j,iua,iub,ju(j),jj,iu(jj),i
      IF(IU(JJ).EQ.I)GO TO 50
      JU(JP)=JJ
c     write(6,*) 'SYMFACT:insideloop50: jp,ju(jp)= '
c     write(6,*) jp,ju(jp)
      JP=JP+1
      IU(JJ)=I
c                  print *,'IU(JJ)=',IU(JJ)
c     write(6,*) 'SYMFACT:insideloop50: jp,jj,iu(jj),i= '
c     write(6,*) jp,jj,iu(jj),i
c     IF(JJ.LT.MIN)MIN=JJ
c                  print *,'chk 2 inside sysfactd'
c                  write(6,*)'chk 2 inside sysfactd'
      IF(JJ.LT.MIN) then
c     write(6,*) 'SYMFACT:insideloop50: jj,min = ',jj,min
      min=jj
c     write(6,*) 'SYMFACT:insideloop50: jj,min = ',jj,min
      endif
   50 CONTINUE
c     write(6,*) 'SYMFACT: jp,jpp,l,last,min,nh = '
c     write(6,*) jp,jpp,l,last,min,nh
c     print *, 'SYMFACT: jp,jpp,l,last,min,nh = '
c     print *, jp,jpp,l,last,min,nh
      IF(JP.EQ.JPP)GO TO 70
      IF(L.NE.LAST)GO TO 40
   60 IF(MIN.EQ.NH)GO TO 90
   70 L=IP(MIN)
c                  print *,'chk 3 inside sysfactd'
c                  write(6,*)'chk 3 insidc sysfactd'
c                  stop
c     write(6,*) 'SYMFACT: min,ip(min),l = '
c     write(6,*) min,ip(min),l
      IF(L.EQ.0)GO TO 80
      IP(I)=IP(L)
c                  print *,'IP(I)=',IP(I)
c     write(6,*) 'SYMFACT: l,ip(l),i,ip(i) = '
c     write(6,*) l,ip(l),i,ip(i)
      IP(L)=I
c     write(6,*) 'SYMFACT: l,ip(l),i,ip(i) = '
c     write(6,*) l,ip(l),i,ip(i)
      GO TO 90
   80 IP(MIN)=I
c                  print *,'IP(MIN)=',IP(MIN)
c                  print *,'chk 4 inside sysfactd'
      IP(I)=I
c     write(6,*) 'SYMFACT: min,ip(min),i,ip(i) = '
c     write(6,*) min,ip(min),i,ip(i)
   90 IU(I)=JPI
c                  print *,'IU(I)=',IU(I)
      IU(N)=JP
      IU(NH)=JP
c                  print *,'IU(N)=',IU(N)
c                  print *,'IU(NH)=',IU(NH)
c     write(6,*) 'SYMFACT: i,iu(i),jpi,n,iu(n),jp,nh,iu(nh) = '
c     write(6,*) i,iu(i),jpi,n,iu(n),jp,nh,iu(nh)
c                  print *,'chk 5 inside sysfactd'
c                  write(6,*)'chk 5 inside sysfactd'
c                  stop
      ncoef2=iu(n+1)-1
c      print *,'iu(n+1) = ',iu(n+1)
c                  print *,'ncoef2 inside symfact =',ncoef2
c                  write(6,*)'ncoef2=',ncoef2
c                  stop
c      write(6,*) 'inside symfact: ncoef2= ', ncoef2
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine transad(n,m,ia,ja,iat,jat)
c      call transad(ndof,ndof,iu,ju,iut,jut)
c      implicit real*8(a-h,o-z)
      dimension ia(*),ja(*),iat(*),jat(*)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c......PLEASE direct your questions to Dr. Nguyen (nguyen@cee.odu.edu)
c......purposes: After symbolic factorization,ja is just a merge lists
c......        WITHOUT ordering (i.e. the nonzero column numbers of a
c......        particular row are 12, 27, 14, 46, 22, 133). Upon
c......        completion, this routine will rearrange the nonzero column
c......        numbers WITH ordering, to be ready for numerical factorization
c......        phase (i.e. 12, 14, 22, 27, 46, 133)
c    input:  ia,ja,an    given matrix in RR(C)U.
c           n           number of rows of the matrix.
c           m           number of columns of the matrix.
c    output: iat,jat,ant  transposed matrix in RR(C)O.
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c      dimension ia(6), ja(7), iat(6), jat(7)
c      dimension ia(neq+1), ja(ncoef1), iat(neq+1), jat(ncoef1)
c      neq=5
c      ncoef1=7
c      nrows=5
c      ncols=5
c......     x  2  0  4  5   ---> input unordered col # 2,5,4
c......        x  3  4  0   ---> input unordered col # 4,3
c......           x  0  5   ---> input col # 5
c......              x  5   ---> input col # 5
c......                 x
c      ia(1)=1
c      ia(2)=4
c      ia(3)=6
c      ia(4)=7
c      ia(5)=8
c      ia(6)=8
c
```

```
c    ja(1)=2
c    ja(2)=5
c    ja(3)=4
c    ja(4)=4
c    ja(5)=3
c    ja(6)=5
c    ja(7)=5
c
c    call transad(nrows,ncols,ia,ja,iat,jat)
c    write(6,*) 'after reordered, ia(-)= ',(ia(i),i=1,neq+1)
c    write(6,*) 'after reordered, ja(-)= ',(ja(i),i=1,ncoef1)
c    stop
c    end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c
c    print *,'ja at start of transd = ',(ja(i),i=1,15)
c                 print *,'check start transd'
c                 print *,'M=',m
c                 print *,'N=',n
c                 print *,'ja start transd=',(ja(ic),ic=1,30)
c                 pause

     MH=M+1
     NH=N+1
     DO 10 I=2,MH
  10 IAT(I)=0

     IAB=IA(NH)-1
c                 print *,'IAB=',iab

     DO 20 I=1,IAB
     J=JA(I)+2
     IF(J.LE.MH)IAT(J)=IAT(J)+1
  20 CONTINUE

c                 print *,'Check 1 inside transd'

     IAT(1)=1
     IAT(2)=1
     IF(M.EQ.1)GO TO 40

     DO 30 I=3,MH

c                 print *,'IAT(_)=',iat(I)
c                 pause

  30 IAT(I)=IAT(I)+IAT(I-1)

  40 DO 60 I=1,N
     IAA=IA(I)
     IAB=IA(I+1)-1
c                                  print *,'Check 2 inside transd'

     IF(IAB.LT.IAA)GO TO 60
     DO 50 JP=IAA,IAB
c                  print *,'IAA=',iaa
c                  print *,'IAB=',iab
c                  print *,'JP=',jp
c                  print *,'JA(JP)=',ja(jp)
     J=JA(JP)+1
c                 print *,'J=',j
     K=IAT(J)
c                 print *,'K=',k
     JAT(K)=I
c                                  print *,'Check 3 inside transd'
c                   print *,'JAT(K)=',jat(k)

c    ANT(K)=AN(JP)
  50 IAT(J)=K+1
c                 print *,'Check 4 inside transd'
c                 print *,'IAA=',iaa
c                 print *,'IAB=',iab
c                 pause
  60 CONTINUE
c                 print *,'Check 5 inside transd'

c    print *,'ja at end of transd = ',(ja(i),i=1,15)
c                 pause
c                 print *,'check before transa2d'
     call transa2d(n,m,iat,jat,ia,ja)
     return
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c=================================================
=======
c    The following routines are extracted from file part2.f
c    see cd ~/cee/newfem/*completely*/
c=================================================
=======

c    Table 8.7  Transposing a Sparse Matrix

c@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@
c..... file = part2.o  (only object codes) ...............
     subroutine transa2d(n,m,ia,ja,iat,jat)
     dimension ia(*),ja(*),iat(*),jat(*)
c                  dimension ja2(20)
c
c                 print *,'ia(_) inside transa2d=',(ia(i),i=1,15)
c                 print *,'ja(_) inside transa2d=',(ja(i),i=1,15)
c                 pause

     MH=M+1
     NH=N+1
     DO 10 I=2,MH
  10 IAT(I)=0

     IAB=IA(NH)-1

     DO 20 I=1,IAB
     J=JA(I)+2
     IF(J.LE.MH)IAT(J)=IAT(J)+1
  20 CONTINUE

     IAT(1)=1
     IAT(2)=1
     IF(M.EQ.1)GO TO 40
     DO 30 I=3,MH
  30 IAT(I)=IAT(I)+IAT(I-1)

c                 Do 31 I=1,20
c   31            ja2(i)=ja(i)

c                 print *,'ja(_) inside transa2d mid=',(ja(i),i=1,10)
c                 pause

  40 DO 60 I=1,N
     IAA=IA(I)
     IAB=IA(I+1)-1
     IF(IAB.LT.IAA)GO TO 60

     DO 50 JP=IAA,IAB
c                 print *,'JA(JP)=',JA(JP),'(JP)',jp
     J=JA(JP)+1
     K=IAT(J)
     JAT(K)=I
c    ANT(K)=AN(JP)
  50 IAT(J)=K+1
c                 print *,'JA(JP) 2=',JA(JP),'(JP)',jp
  60 CONTINUE
c                 print *,'JA(JP) 3=',(JA(k),k=1,20)
c
     nz=iat(m+1)-1
c     write(6,*) 'inside transa2: iet(-) = ',(iat(i),i=1,m+1)
c     write(6,*) 'inside transa2: jet(-) = ',(jat(i),i=1,nz)
c     print *, 'inside transa2 at end: iet(-) = ',(iat(i),i=1,m+1)
c     print *, 'inside transa2 at end: jet(-) = ',(jat(i),i=1,nz)
c                 stop
c                 Do 61 i=1,20
c   61            ja(i)=ja2(i)

c                 print *,'ja(_) inside transa2d at end=',(ja(i),i=1,10)
c                 pause

     return
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

c    Table 8.2  Complete FORTRAN Code for Sparse, Symmetrical
c            Symbolic Assembly Process

     subroutine symbass(ie,je,iet,jet,ndof,ia,ja,ncoef1,istep)
c     subroutine symbass(ie,je,iet,jet,ndof,ia1,ia2,ja,ncoef1)
     implicit real*8(a-h,o-z)
     dimension ie(*),je(*),iet(*),jet(*),ja(*),ia(*)
c     dimension ie(*),je(*),iet(*),jet(*),ja(*),ia1(*),ia2(*)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c......PLEASE direct your questions to Dr. Nguyen (nguyen@cee.odu.edu)
c......Purposes: symmetrical, sparse symbolic assembly
c......       This code is stored under file name *symb*.f, in sub-directory
c......       cd ~/cee/newfem/complete*/part2.f
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c......Input: ie(nel+1)=locations (or pointers) of the first non-zero
c......     of each row (of element-dof connectivity info.)
c......      je(nel*ndofpe)=global dof column number for non-zero
c......      terms of each row (of element-dof connectivity info.)
c......     iet(ndof+1)=locations (pointers) of the first non-zero
c......     of each row (of dof-element connectivity info.)
c......      jet(nel*ndofpe)=locations (pointers) of the first non-zero
c......      of each row (of dof-element connectivity info.)
c......     ia(ndof)= ndof in the positions corespond to Dirichlet b.c.
c......           0  elsewhere
c......Output:ia(ndof+1)=starting locations of the first non-zero
c......     off-diagonal terms for each row of structural stiffness
c......     matrix
c......      ja(ncoeff)=column numbers (unordered) corespond to
c......      each nonzero, off-diagonal term of each row of structural
c......      stiffness matrix
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c "ia1" contains the supportdof from subroutine supportdof
c "ia2" is the new "ia" that gets set to zero at the start of this
c  subroutine and is created new at the end. GAH 4-23-04
c     print *,'*** ***  START SYMBASS  *** ***'
c                 PAUSE
c     print *,'*** ***  START SYMBASS  *** ***'
c                 PAUSE
c     print *,'*** ***  START SYMBASS  *** ***'
c                 PAUSE
c     print *,'*** ***  START SYMBASS  *** ***'
c                 PAUSE
c     print *,'ia(-) array at start symbass = ',(ia(i),i=1,ndof+1)
c                 PAUSE
c                  stop

                  n=ndof
     jp=1                     !001
     nm1=n-1                   !002
c                 print *,'nm1=',nm1
c                 print *,'n=',n
c                 print *,'ndof=',ndof
c                  print *,'ie(i) inside symbass=',(ie(i),i=1,nm1)
c                  print *,'je(i) inside symbass=',(je(i),i=1,nm1)
c                 stop
c                  print *,'ja(i) inside symbass=',(ja(i),i=1,nm1)
c                  print *,'jet(i) inside symbass=',(jet(i),i=1,nm1)
c                  print *,'iet(i) inside symbass=',(iet(i),i=1,nm1)
c                  print *,'ia1(i) start symbass=',(ia1(i),i=1,ndof+1)
c                  print *,'ia2(i) start symbass=',(ia2(i),i=1,ndof+1)
c                 pause
c initialize to zero
```

```
c                   do 9 k=1,ndof+1
c                   ia1(k)=0.
c                   ia2(k)=0.

c9                  continue
c                   stop
      do 30 i=1,nm1                  !003 last row (= eq) will be skipped
      jpi=jp                    !004 delayed counter for ia(-) array
c      if ( ia2(i) .eq. n ) go to 30     !005 skip row which corespond to Dirichlet b.c.
      if ( ia(i) .eq. n ) go to 30     !005 skip row which corespond to Dirichlet b.c.

      ieta=iet(i)                    !006 begin index (to find how many elements)
      ietb=iet(i+1)-1                  !007 end index (to find how many elements)
       do 20 ip=ieta,ietb             !008 loop covering ALL elements attached to row i
       j=jet(ip)                    !009 actual "element number" attached to row i
       iea=ie(j)                    !010 begin index (to find how many nodes attached to
element j)
c                    print *,'ie(j)inside symbass=',ie(j)
c      print *,'ie(j+1)=',ie(j+1)
       ieb=ie(j+1)-1                  !011 end index (to find how many nodes attached to
element j)
c                   print *,'iea & ieb=',iea,ieb
        do 10 kp=iea,ieb              !012 loop covering ALL nodes attached to element j
        k=je(kp)                    !013 actual "node, or column number" attached to element j
c                    print *,'k in symbass=',k
c                    print *,'i=',i
c                    print *,'ia2(k)=',ia2(k)
c                   pause
        if (k .le. i) go to 10       !014 skip, if it involves with LOWER triangular portion
        if ( ia(k) .ge. i ) go to 10  !015 skip, if same node already been accounted by earlier
elements
c          if ( ia2(k) .ge. i ) go to 10  !015 skip, if same node already been accounted by
earlier elements

        ja(jp)=k                    !016 record "column number" associated with non-zero off-
diag. term
        jp=jp+1                     !017 increase "counter" for column number array ja(-)
        ia(k)=i                    !018 record node (or column number) k already contributed
to row i
c          ia2(k)=i                   !018 record node (or column number) k already
contributed to row i

c                     print *,'ia2(k)=',ia2(k)
 10       continue                   !019
 20     continue                     !020
c      print *,'jpi = ',jpi
c                    pause
 30   ia(i)=jpi                      !021 record "starting location" of non-zero off-diag.
c 30   ia2(i)=jpi                      !021 record "starting location" of non-zero off-diag.

c                           !   terms associated with row i
c                    print *,'jp inside symbass=',jp
c                   pause
      ia(n)=jp                       !022 record "starting location" of non-zero term of LAST
ROW
      ia(n+1)=jp                      !023 record "starting location" of non-zero term of LAST
ROW + 1
c      ia2(n)=jp                       !022 record "starting location" of non-zero term of LAST
ROW
c      ia2(n+1)=jp                      !023 record "starting location" of non-zero term of LAST
ROW + 1

c
      ncoef1=ia(n+1)-1
c      ncoef1=ia2(n+1)-1

c                   If(istep .eq. 1)then
c                   print *,'ia(n)=',ia(n)
c                   print *,'ia(n+1)=',ia(n+1)
c                   print *,'ncoef1 inside symbass=',ncoef1
c                   pause
c                   endif

c                   pause
c                   print *,'ia(_) at end of symbass=',(ia(i),i=1,ndof+1)
c                   pause
c      write(6,*) 'ia(-) array = ',(ia(i),i=1,n+1)
c      write(6,*) 'ja(-) array = ',(ja(i),i=1,ncoef1)
c
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c      Table 8.4  Complete FORTRAN Code for Sparse, Symmetrical
c              Numerical Assembly Process

      subroutine numass(ia,ja,idir,ae,be,lm,ndofpe,an,ad,b,ip,
     $elmass,dm,am,ianal,lumpmass,ndof,ncoef1,inc,istep)
c      subroutine numass(ia2,ja,idir,ae,be,lm,ndofpe,an,ad,b,ip,
c     $elmass,dm,am,ianal,lumpmass,ndof,ncoef1)
      implicit real*8(a-h,o-z)
c       dimension ia(*),idir(*),ae(*),be(*),lm(*),an(*)
      dimension ia(*),ja(*),idir(*),ae(*),be(*),lm(1),an(*)
c       dimension ia(*),idir(*),ae(*),be(*),lm(*),an(*)
c       dimension ia2(*),ja(*),idir(*),ae(*),be(*),lm(*),an(*)
      dimension ad(*),b(*),ip(*)
      dimension elmass(1),dm(1),am(1)
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
c......PLEASE direct your questions to Dr. Nguyen (nguyen@cee.odu.edu)
c......Purposes: symmetrical, sparse numerical assembly
c......          This code is stored under file name *symb*.f, in sub-directory
c......          ~
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
c......Input: ia(ndof+1)=starting locations of the first non-zero
c......      off-diagonal terms for each row of structural stiffness
c......      matrix
c......       ja(ncoeff)=column numbers (unordered) corespond to
c......        each nonzero, off-diagonal term of each row of structural
c......        stiffness matrix
c......      idir(ndof)= 1 in the positions corespond to Dirichlet b.c.
c......              0 elsewhere
c......       ae(ndofpe**2),be(ndofpe)= element (stiffness) matrix,
c......       and element (load) vector
c......      lm(ndofpe)= global dof associated with a finite element
c......       ndofpe= number of dof per element
c......      b(ndof)= before using this routine, values of b(-) should
c......      be initialized to:
c......      Ci, values of prescribed Dirichlet bc at proper locations
c......      or  values of applied nodal loads
c
c......Output:  an(ncoeff1)= values of nonzero, off-diagonal terms of
c......        structural stiffness matrix
c......      ad(ndof)= values of diagonal terms of structural stiffness
c......      matrix
c......       b(ndof)= right-hand-side (load) vector of system of linear
c......       equations
c......Temporary Arrays:
c......      ip(ndof)= intialized to 0
c......            then IP(-) is used and reset to 0
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
c                   print *,'***** chk start numass *****'
c                   pause
c                   print *,'***** chk start numass *****'
c                   pause
c                   print *,'***** chk start numass *****'
c                   pause

c                   If(istep .eq. 1)then
c                   print *,'***** chk start numass *****'
c                   pause

c                   print *,'lm(i) start numass=',(lm(i),i=1,ndofpe)
c                   pause
c                   print *,'ndofpe inside numass =',ndofpe
c                   print *,'ndof inside numass=',ndof

c                   print *,'ad diag stiff start numass=',(ad(i),i=1,ndof)
c                   print *,'ae off diag stiff start numass=',(ae(i),i=1,ncoef1)
c      print *, 'an(i) array start numass = ',(an(i),i=1,50)
c      pause
c                   print *,'ia(-) start of numass = ',(ia(i),i=1,15+1)
c                   print *,'ja(-) start in numass = ',(ja(i),i=1,15+1)
c                   pause
c                   endif

c                   stop
c                   print *,'ae start of numass =',(ae(i),i=1,16)
C                   stop
c                   print *,'an start on numass =',(an(i),i=1,20)
c                   print *,'ad start on numass =',(ad(i),i=1,20)
c                   print *,'ip start in numass =',(ip(i),i=1,20)
c                   pause

c                   print *,'ia2(-) start of numass = ',(ia2(i),i=1,9+1)
c                   stop
c      print *,'ndofpe=',ndofpe
c                   print *,'idir=',(idir(ic),ic=1,ndof)
c                   pause

c      print *, 'ia2(-) array = ',(ia2(i),i=1,ndof+1)
c                   print *,'ncoef1=',ncoef1
c      print *, 'ja(-) array inside numass = ',(ja(ic),ic=1,ncoef1)
c                   print *,'ianal inside numass=',ianal
c                   print *,'idir=',(idir(i),i=1,ndof)
c                   print *,'b at start numass=',(b(i),i=1,ndof)
c                   pause
c                   print *,'be in numass=',(be(i),i=1,ndofpe)
c                   print *,'ia2=',(ia2(i),i=1,ndof)
c                   print *,'an=',(an(i),i=1,30)
c                   stop
c      write(*,*) 'ia2(-) start of numass = ',(ia2(i),i=1,ndof+1)
c      write(*,*) 'inside numass: b(-) = ',(b(i),i=1,10)
c                   stop
c      write(6,*) 'ia2(-) start of numass = ',(ia2(i),i=1,ndof+1)
c                   print *,'ia2(i) start numass=',(ia2(i),i=1,ndof+1)
c                   pause
c                   stop
c      write(6,*) 'inside numass: b(-) = ',(b(i),i=1,10)
      do 40 L=1,ndofpe                 !001 local "row" dof
      i=lm(L)                      !002 global"row" dof
c       print *,'lm(L)=',lm(L),'  L=',L
c                    print *,'i=',i
c                    print *,'idir(i) in numass=',idir(i)
c                    pause
      if ( idir(i) .ne. 0 ) go to 401    !003 skip, if DIRICHLET b.c.
c       print *,'*** idir=0 ***'
                    k=L-ndofpe                    !004 to find location of element k-diag
c                    print *,'L=',L
c                    print *,'ndofpe=',ndofpe
c                    print *,'k=L-ndofpe',k
c                   print *,'ae(k+L*ndofpe)=',ae(k+L*ndofpe)
c                   pause
      ad(i)=ad(i)+ae(k+L*ndofpe)        !005 assemble K-diag

c                   print *,'ad in numass=',ad(i)
c                   pause
       if (ianal .eq. 1) then
       dm(i)=dm(i)+elmass(k+L*ndofpe)    !    assemble M-diag
       endif
c       print *,'assemble element rhs Load Vector in numass'
      b(i)=b(i)+be(L)                 !006 assemble element rhs load vector
c       print *,'be(L) = ',be(L)
c                    print *,'b(i) = ',(b(ic),ic=1,ndof)
c                   stop
c                   pause

      kk=0                          !007 flag, to skip contribution of entire
c                             !   global row i if all global col #
c                             !   j < i, or if entire row i belongs
c                             !   to LOWER triangle
```

```
      do 20 LL=1,ndofpe              !008 local "column" dof
c                    print *,'i two=',i
c                    print *,'ndofpe=',ndofpe
c                      print *,'LL=',LL
c                   pause
      k=k+ndofpe                    !009 find location of element stiffness k
c                   print *,'ndofpe=',ndofpe
c                   print *,'k=',k

c                   stop
      if (LL .eq. L) go to 20        !010 skip, diag. term already taken care
      j=lm(LL)                      !011 global column dof
c                    print *,'lm(LL)=',lm(LL)
c                    print *,'i three=',i
c                     print *,'j=',j
c                   pause
c                   print *,'idir(j)=',idir(j)
c                   pause
      if ( idir(j) .ne. 0 ) go to 10   !012 skip, if DIRICHLET b.c.
      if (j .lt. i) go to 20         !013 skip, if LOWER portion
      ip(j)=k                       !014 record global column # j (associated with
c                          !   global row # i) corespond to k-th term
c                      print *,'ip(j)=',ip(j)
c                      print *,'j=',j
c                      print *,'i=',i
c                      pause
c                      write(6,*)'ip(j)=',ip(j)
c                      write(6,*)'j=',j
c                      write(6,*)'i=',i
c                   pause

c                   stop
c                          !   of element stiffness k
      kk=1                          !015 FLAG, indicate row L of [k] do have
c                          !   contribution to global row I of [K]
c                   print *,'kk in numass=',kk
c                   pause
      go to 20                      !016
 10    b(i)=b(i)-b(j)*ae(k)          !017 modify rhs load vector due to DIRICHLET b.c.

c                    print *,'ae(k)=',ae(k)
c                    print *,'i four=',i
c                    print *,'LL=',LL
c                   pause
c                    print *,'b(j)=',(b(j),j=1,ndof)

c                   stop
 20    continue
c                   print *,'Finished Do 20 in numass'
c                    print *,'b(i)=',(b(ic),ic=1,ndof)
c                    write(6,*)'kk=',kk
c                   print *,'kk=',kk
c                    stop
c                    print *,'L=',L
c                    print *,'i five=',i
c                   pause
     if (kk .eq. 0) go to 40         !019 skip indicator (see line 007)
     iaa=ia(i)                      !020 start index
     iab=ia(i+1)-1                  !021 end index
c                    write(6,*)'iaa=',iaa
c                    write(6,*)'iab=',iab
c                   print *,'i =',i
c                    print *,'ia(i)=',ia(i)
c                    print *,'ia2(i+1)=',ia2(i+1)
c                    print *,'iaa=',iaa
c                    print *,'iab=',iab
c                   pause
c                   stop
      do 30 j=iaa,iab               !022 loop covering all col numbers associated
c                          !   with global row i
c                    print *,'j=',j
c                    print *,'ja(j) = ',ja(j)
c                   stop
c                    pause
      k=ip( ja(j) )                 !023 ip ( col # ) already defined on line 014
c                    print *,'k=',k

c                    write(6,*)'ja(j)=',ja(j)
c                    print *,'ja(j)=',ja(j)

c                    write(6,*)'k=',k
c                    print *,'ja(j)=',ja(j)

c                   pause
c                   print *,'j=',j
c                   print *,'an(j)=',an(j)
c                   print *,'iaa=',iaa
c                   print *,'iab=',iab
c                   stop
c                   pause
c                          !            or initialized to ZERO initially
      if (k .eq. 0) go to 30        !024 skip, because eventhough row L of [k] do
c                          !   have contribution to global row I of {K},
c                          !   some terms of row L (of [k]) which associated
c                          !   with DIRICHLET b.c. columns should be SKIPPED
c                   print *,'########## an & ac ##########'
c                   pause
c                   print *,'ae(k) in numass before=',ae(k)
c                   print *,'an(j) in numass before=',an(j)
c                   pause
      an(j)=an(j)+ae(k)             !025 assemble [K] from [k]

c                     write(6,*)'an(j)=',an(j)
c                     write(6,*)'ae(k)=',ae(k)
c     print *,'ae(k) in numass =',ae(k)
c                   print *,'######### an #########'
c                   pause
c                   print *,'j in numass=',j
c                    print *,'an in numass=',an(j)
c                   pause
c                    stop
      if (ianal .eq. 1  .and.  lumpmass .eq. 0) then
      am(j)=am(j)+elmass(k)          !   assemble off-diag. [M] from [m]
      endif
      ip( ja(j) )=0                 !026 reset to ZERO for col # j before considering
c                          !   the next row L
 30     continue                    !027
     go to 40
 401  continue
     ad(i)=1.0                      !   reset the diagonal of K(i,i)=1.0, due to b.c.
     if (ianal .eq. 1) dm(i)=1.0      !   reset the diagonal of M(i,i)=1.0, due to b.c.
c     write(6,*) 'inside numass: dirichlet dof#, ad(-) = ',i, ad(i)
c                    print *,'inside numass: dirichlet dof#, ad(-) = ',i, ad(i)
c                   stop
 40   continue                      !028
c     print *,'stop after 40'
c      print *, 'ad(i) diag stiff= ',(ad(ic),ic=1,ndof)
c      print *, 'an(i) off diag stiff = ',(an(ic),ic=1,50)
c                    pause
c                    stop
c......print debugging results
c     ndof=10
c     ncoef1=ia(ndof+1)-1
c     write(6,*) 'at the end of routine numass'
c     write(6,*) 'ia2(-) array = ',(ia2(i),i=1,ndof+1)
c     write(6,*) 'ja(-) array = ',(ja(i),i=1,ncoef1)
c     write(6,*) 'ad(-) array = ',(ad(i),i=1,ndof)
c     write(6,*) 'an(i) array = ',(an(i),i=1,3)
c     write(6,*) ' b(-) array = ',( b(i),i=1,ndof)
c                   write(6,*) '*** END of numass ***'
c     print *, 'ia(-) array = ',(ia(i),i=1,ndof+1)
c     print *, 'ja(-) array = ',(ja(i),i=1,ncoef1)
c     print *, 'ad(-) array = ',(ad(i),i=1,ndof)
c     print *, 'an(i) array = ',(an(ic),ic=1,50)
c     print *, ' b(-) array end numass= ',( b(ic),ic=1,ndof)
c                   print *,'ad diag stiff end numass=',(ad(ic),ic=1,ndof)
c                   print *,'ae off diag stiff end numass=',(ae(i),i=1,ncoef1)

c                    print *,'at the end of routine numass'
c                   pause
c                    print *,'at the end of routine numass'
c                   pause
c                    print *,'at the end of routine numass'
c                   pause

c                   stop
     return
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
     subroutine materprop(nummat,prop,youngm)
     implicit real*8(a-h,o-z)
     dimension prop(1)
c
c......Purposes: Input material properties, assuming max. 10 materials,
c           and each material type has 11 properties
c......Notes:    This routine can be used for any finite element type !
c
     nprop=11
      do 1 i=1,nummat
      istart=(i-1)*nprop
      read(5,*) e1,e2,e3,xnu12,xnu23,xnu31,g12,g23,g31,rho,thermoc
c       write(6,*) 'e1,e2,e3,xnu12,xnu23,xnu31,g12,g23,g31,rho,thermoc'
c       write(6,*)  e1,e2,e3,xnu12,xnu23,xnu31,g12,g23,g31,rho,thermoc
      prop(istart+1)=e1
      prop(istart+2)=e2
      prop(istart+3)=e3
      prop(istart+4)=xnu12
      prop(istart+5)=xnu23
      prop(istart+6)=xnu31
      prop(istart+7)=g12
      prop(istart+8)=g23
      prop(istart+9)=g31
      prop(istart+10)=rho
      prop(istart+11)=thermoc
 1     continue

                   youngm=e1

c                    print *,'e1=',e1
c                    print *,'e2=',e2
c                   stop

     return
     end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
     subroutine sectprop(nsect,prop,area)
     implicit real*8(a-h,o-z)
     dimension prop(1)
c
c......Purposes: Input cross-sectional properties, assuming max. 10 sections,
c           and each cross-sectional type has 12 properties
c......Notes:    This routine can be used for any finite element type !
c
c      nprop=12
      nprop=1
c                   print *,'nsect inside sectprop=',nsect
c                   pause
      do 1 i=1,nsect
      istart=(i-1)*nprop
cc                  print *,'istart=',istart
c       read(5,*) area,xi,yi,zi,t1,t2,t3,t4,
c     $theta1,theta2,theta3,theta4
      read(8,*) area
      write(6,*) 'area'
      write(6,*)  area
c       write(6,*) 'area,xi,yi,zi,t1,t2,t3,t4'
c       write(6,*) 'theta1,theta2,theta3,theta4'
c       write(6,*)  area,xi,yi,zi,t1,t2,t3,t4
c       write(6,*)  theta1,theta2,theta3,theta4

      prop(istart+ 1)=area
```

```
cc                    print *,'prop(_)=',prop(i)
c       prop(istart+ 2)=xi
c       prop(istart+ 3)=yi
c       prop(istart+ 4)=zi
c       prop(istart+ 5)=t1
c       prop(istart+ 6)=t2
c       prop(istart+ 7)=t3
c       prop(istart+ 8)=t4
c       prop(istart+ 9)=theta1
c       prop(istart+10)=theta2
c       prop(istart+11)=theta3
c       prop(istart+12)=theta4

 1     continue

c                      area = prop(istart+1)

cc                    print *,'area inside sectprop=',area
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine nodecoor(numnodes,xxx,yyy,zzz)
      implicit real*8(a-h,o-z)
c
c......Purpose: Input joint coordinates
c
      dimension xxx(1),yyy(1),zzz(1)
c
c                     print *,'numnodes inside nodecoor=',numnodes
c                     pause

c      write(6,*) 'joint,xxx(joint),yyy(joint),zzz(joint)'
      do 21 i=1,numnodes
      read(5,*) joint,xxx(joint),yyy(joint),zzz(joint)
c                     write(*,4)  joint,xxx(joint),yyy(joint),zzz(joint)
c                    pause
c      write(6,4)  joint,xxx(joint),yyy(joint),zzz(joint)
c 4    format(2x,i5,f10.2,f10.2,f10.2)
 21    continue

      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine elconect(ie,je,lm,neltype,nel,ndofpe,ndofpn,iarea)
      implicit real*8(a-h,o-z)
c
c......Purpose: Input element connectivity information.
c......Note: This routine can be used for any finite element type ??
c
      dimension ie(1),je(1),lm(1),nel(1),ndofpe(1)
                     dimension iarea(*)
c
                     itharea=0

       icadd=0
       locadd=0
      ie(1)=1                  ! always
c                    print *,'neltype inside elconect=',neltype

c                    pause
c                    stop
      do 21 k=1,neltype
      numel=nel(k)
c                     print *,'numel inside elconect=',numel
c                    pause
      if (numel .eq. 0) go to 21
      go to (31,32,33,34,34,34),k
 31    continue                 ! 2-d truss
c                    print *,'numel=',numel
c                    pause
      do 22 i=1,numel
      ie(i+1)=ie(i)+ndofpe(k)
c                    print *,'ie(i+1) inside elconect =',ie(i+1)
c                    pause
c
c      read(5,*) ithel,nodei,nodej
      read(5,*) ithel,itharea,nodei,nodej
cc                   print *,'itharea=',itharea
c                    print *,'ithel inside elconect=',ithel
                     iarea(i)=itharea
c                    print *,'i=',i,' iarea(i)inside elconect=',iarea(i)
c                    print *,'Read inside elconect'
c      write(6,*) 'ithel,nodei,nodej = ',ithel,nodei,nodej
c      print *,'ithel,nodei,nodej = ',ithel,nodei,nodej
c                    pause
c Use for 2D truss
c      lm(1)=nodei*ndofpn-1
c      lm(2)=lm(1)+1
c      lm(3)=nodej*ndofpn-1
c      lm(4)=lm(3)+1
c                     print *,'nodei=',nodei
c                     print *,'ndofpn=',ndofpn
c Use below for 3D Truss
      lm(1)=nodei*ndofpn-2
      lm(2)=lm(1)+1
                     lm(3)=lm(1)+2
      lm(4)=nodej*ndofpn-2
      lm(5)=lm(4)+1
      lm(6)=lm(4)+2
      locbegin=(ithel-1)*ndofpe(k)
c                     print *,'ithel=',ithel
c                     print *,'nodei=',nodei
c                     print *,'nodej=',nodej
c                     print *,'ndofpn=',ndofpn
c                     print *,'lm(1)=',lm(1)
c                     print *,'lm(2)=',lm(2)
c                     print *,'lm(3)=',lm(3)
c                     print *,'lm(4)=',lm(4)
c                     print *,'lm(5)=',lm(5)
c                     print *,'lm(6)=',lm(6)
c                     print *,'locbegin=',locbegin
c                     pause
       do 23 j=1,ndofpe(k)
       locate=locbegin+j
       je(locate)=lm(j)
c                     print *,'locate=',locate
c                     print *,'je(locate)=',je(locate)
c                     pause
c                        print *,'lm(j)=',lm(j)
c                        print *,'je(locate)=',je(locate)
c        pause

 23     continue
 22    continue
      go to 21
 32    continue                 ! 3-d beam
      ieadd=ieadd+nel(k-1)        ! to include previous 2-d truss elements
      locadd=locadd+nel(k-1)*4
      go to 21
 33    continue                 ! 4-node (24 dofpe) rect. plate bending+membrane
      ieadd=ieadd+nel(k-1)        ! to include previous 3-d beam elements
      locadd=locadd+nel(k-1)*12
c                     print *,'numel=',numel
c                     pause
      do 42 i=1,numel
      ie(i+1+icadd)=ie(i+icadd)+ndofpe(k)
c
      read(5,*) ithel,nodei,nodej,nodck,nodel
      write(6,*) 'ithel,nodei,nodej,nodek,nodel'
      write(6,*)  ithel,nodei,nodej,nodek,nodel
c
      lm(1)=nodei*ndofpn-5
      lm(2)=lm(1)+1
      lm(3)=lm(2)+1
      lm(4)=lm(3)+1
      lm(5)=lm(4)+1
      lm(6)=lm(5)+1
c
      lm( 7)=nodej*ndofpn-5
      lm( 8)=lm( 7)+1
      lm( 9)=lm( 8)+1
      lm(10)=lm( 9)+1
      lm(11)=lm(10)+1
      lm(12)=lm(11)+1
c
      lm(13)=nodek*ndofpn-5
      lm(14)=lm(13)+1
      lm(15)=lm(14)+1
      lm(16)=lm(15)+1
      lm(17)=lm(16)+1
      lm(18)=lm(17)+1
c
      lm(19)=nodel*ndofpn-5
      lm(20)=lm(19)+1
      lm(21)=lm(20)+1
      lm(22)=lm(21)+1
      lm(23)=lm(22)+1
      lm(24)=lm(23)+1
c
      locbegin=(ithel-1)*ndofpe(k)
       do 43 j=1,ndofpe(k)
       locate=locbegin+j
       je(locate+locadd)=lm(j)
 43     continue
 42    continue
      go to 21
 34    continue                 ! other f.e. type
      go to 21

 21    continue
c
      ieall=nel(1)+nel(2)+nel(3)+nel(4)+nel(5)+nel(6)
cc      jeall=nel(1)*4 + nel(2)*12 + nel(3)*24
      jeall=nel(1)*6 + nel(2)*12 + nel(3)*24
c                    print *,'jeall=',jeall
c                    pause
c
c      write(6,*) 'ieall = sum all el types = ',ieall
c      write(6,*) 'ie= ', (ie(i),i=1,ieall+1)
c      write(6,*) 'je = ',  (je(i),i=1,jeall)
c      print *, 'ieall = sum all el types = ',ieall
c                    print *, 'jeall = ',jeall
c                    print *, 'ie= ', (ie(i),i=1,ieall+1)
c                    print *,'iarea(_)=',(iarea(i),i=1,numel)
c      print *, 'je = ',  (je(i),i=1,jeall)
c                    pause
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      subroutine loads(ndof,b,loadof,exfor)
      implicit real*8(a-h,o-z)
c
c......Purpose: Input applied loads at the joints
c
      dimension b(1),exfor(*)
c
c       write(6,*) 'inside loads: ndof = ',ndof
c                      print *,'inside loads'
      do 1 i=1,ndof
                    exfor(i)=0.0
      b(i)=0.0    ! load vector
 1     continue

       do 18 i=1,loadof
       read(5,*) loadedof, value
c        write(6,*) 'loadedof, value'
c        write(6,*)  loadedof, value

c                    print *,'loadof=',loadof
c                    print *,'value=',value
c                    pause
```

```
      b(loadedof)=value
                  exfor(loadedof)=value
 18   continue
c                 print *,'exfor inside loads=',(exfor(i),i=1,ndof)
c                 stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      subroutine limits(ndof,numlim,dlimit,ilimit)
      implicit real*8(a-h,o-z)
c
c......Purpose: Reads in constraints, number, dof & max allow disp
c
      dimension dlimit(*),ilimit(*)
c
      do 1 i=1,ndof
                 ilimit(i)=0.0
     dlimit(i)=0.0
 1    continue

      do 18 i=1,numlim !numlim is the number of optimizer constraints
      read(5,*) limitdof, ivalue, dvalue
c       write(6,*) 'loadedof, value'
c       write(6,*)  loadedof, value
c                 print *,'loadof=',loadof
c                 print *,'value=',value
c                 pause
      ilimit(limitdof)=ivalue !dof of optimizer constraint
                  dlimit(limitdof)=dvalue !value of optimizer constraint
 18   continue
c                 print *,'exfor inside loads=',(exfor(i),i=1,ndof)
c                 stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      subroutine supportdof(iboundc,nboundc,b,ndof,ia)
      implicit real*8(a-h,o-z)
c
c......Purpose: input support dof information
c
      dimension iboundc(1),       b(1),ia(1)
      do 18 i=1,ndof
      iboundc(i)=0
 18   continue
c                 print *,'nboundc inside supportdof=',nboundc
c......input the boundary condition flag
       do 19 i=1,nboundc
       read(51,*) ibcdof, settlem
c        write(6,*) 'ibcdof, settlem = ',ibcdof,settlem
       iboundc(ibcdof)=1
       b(ibcdof)=settlem
c                 print *,'b(i) inside supportdof=',b(i)
 19     continue
c                 pause
c     write(6,*) ' ibcdof = ', ibcdof
c      write(6,*) ' load and settlement b(-) = ', (b(i),i=1,ndof)
c......copy boundary flag for later usage in Duc's sparse symbolic assembly
      do 3 i=1,ndof
      if ( iboundc(i) .eq. 0 ) ia(i)=0
      if ( iboundc(i) .eq. 1 ) ia(i)=ndof
cc                print *,'ia(_) inside supportdof=',ia(i)
 3    continue

                  rewind 51

cc                pause
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      subroutine assembly(ip,ad,an,lm,ie,je,propmat,
c    $         propsect,elk,be,b,iboundc,
     $        propsect,be,b,iboundc,iarea,
     $        ae,ia,ja,xx,yy,zz,ncoef1,maxdofpe,
c    $         ae,ia,ja,ncoef1,maxdofpe,
c    $         ae,ia2,ja,xx,yy,zz,ncoef1,maxdofpe,
     $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c    $ tempo1,disp,iarc,it,area,youngm,esm,qk,itnrmlon)
     $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon,istep,icheck,
     $      strainel,strainE,strainD,weight,Anew,Apart,Fin1,
     $      Fin2,VED,VEDold,xlie,SED)

      implicit real*8(a-h,o-z)
c
c                 dimension exfor(*)

c     Purpose: perform sparse numerical assembly
c
c
      dimension ip(1),ad(1),an(1),lm(1),ie(1),je(1),propmat(1)
     $,      propsect(1),be(1),b(1),iboundc(1),iarea(1)
c    $,       propsect(1),elk(maxdofpe,maxdofpe),be(1),b(1),iboundc(1)
     $,   ae(1),ia(1),ja(1),xx(500),yy(500),zz(500),nel(1),ndofpe(1)
c    $,       ae(1),ia2(1),ja(1),xx(1),yy(1),zz(1),nel(1),ndofpe(1)
      dimension dm(1),am(1),elm(maxdofpe,1),tempo1(1)

     common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
     common/disp/ux(2),uy(2),uz(2),P(6),B1(6)

                 common/jam/ja2(2000)
c     common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)

c                 dimension ux(2),uy(2),finx(2),finy(2)
                  dimension disp(*),qk(*)

                  dimension lm2(6)

                  dimension elk(6,6)

                  dimension strainel(*),strainE(*),strainD(*),weight(*),xlie(*)

                  dimension Anew(*),Apart(*),Fin1(*),Fin2(*),VED(*),VEDold(*),SED(*)

c                 print *,'je start assembly=',(je(ic),ic=1,10)
c                 pause
c                 print *,'B(_) start assembly=',(b(ic),ic=1,ndof)
c                 pause
c                 print *,'Start assembly'
c                 print *,'ux(1)=',ux(1)
c                 print *,'uy(1)=',uy(1)
c                 print *,'uz(1)=',uz(1)
c                 print *,'ux(2)=',ux(2)
c                 print *,'uy(2)=',uy(2)
c                 print *,'uz(2)=',uz(2)
c                 pause

c     double precision, allocatable:: b(:,:)
c                 allocate elk(maxdofpe,maxdofpe)

c                  dimension ja(1)

c                 print *,'Getstiff start assembly=',getstiff
c                 print *,'Getforce start assembly=',getforce
c                 print *,'Getsolve start assembly=',getsolve
c                 pause

                  If(getsolve .eq. 1)Then
                  Goto 670
                  Endif

                  If(getforce .eq. 1)Then !Skip ad & ae =0,just getforce
                  Goto 699
                  Endif

c                 print *,'S T A R T  O F  A S S E M B L Y'
c                 print *,'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
c                 print *,'Setting ad, an & ae to zero'
      Do 701 ic=1,ndof
                   ad(ic)=0.0 !Stiffness diagonal

c                  an(ic)=0.0 !Stiffness off diagonal

701   continue

                   Do 702 ic=1,10000
                   an(ic)=0.d0
702   continue


c                 print *,'b(_) at start2 of assembly=',(b(ic),ic=1,ndof)
c                 pause
c                  Do 702 ic=1,100
c                  ae(ic)=0.0
c702     continue
c                 print *,'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
c                 pause
699               Continue

c                 print *,'idir at start of assembly=',(idir(ic),ic=1,ndof)
c                 pause

c                 print *,'qk(_) at start of assembly=',(qk(ic),ic=1,ndof)
c                 print *,'b(_) at start of assembly=',(b(ic),ic=1,ndof)
c                 pause
c                 print *,'ia start of assembly=',(ia(i),i=1,15)
c                 print *,'ja2(i)=',(ja2(ic),ic=1,10)
c                 print *,'ja(1)=',ja(1)
c                 print *,'ja start of assembly=',(ja(i),i=1,1)
c                 print *,'ndofpe = ',ndofpe
c                 stop

c                 If(getforce .eq. 1)then
c                 Do 777 i = 1,10
c                 ja(i)=ja2(i)
c                 Print *,'ja2(i) inside start assembly=',ja2(i)
c777              Continue
c                 pause
c                 Endif

c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 print *,'ianal=',ianal
c                 stop
c                 endif
c                 print *,'getforce=',getforce
c                 stop
c                 print *,'chk inside assembly at start'
c                 print *,'getstiff=',getstiff
c                 print *,'getforce=',getforce
c                 pause
c                 print *,'xx inside assembly = ',(xx(i),i=1,42)
c                 print *,'yy inside assembly = ',(yy(i),i=1,42)
c                 print *,'zz inside assembly = ',(zz(i),i=1,42)
c                 print *,'lm inside assembly =',(lm(i),i=1,6)
c                 print *,'_______________________________________'
c                 pause
c                 stop
c
c                 pause

c                  If(getforce .eq. 1)then
c                  print *,'getforce=1 at start of assembly'
c                  stop
c                  endif

c                 write(6,*)'ia2 start of assembly=',(ia2(i),i=1,ndof+1)
c                  print *,'chk inside assembly'
c                 print *,'iarc inside assembly=',iarc
c                 print *,'ianal inside assembly=',ianal
c                 print *,'neltype=',neltype
c                                 print *,'ncoef1=',ncoef1
```

```
c                 print *,'ndof=',ndof
cc                ncoef2=ncoef1
c                 print *,'ip=',(ip(i),i=1,ndof)
c                 stop
c                 print *,'b=',(b(i),i=1,ndof)
c                 pause
c                  print *,'ia2 start assembly=',(ia2(i),i=1,ndof+1)
c                 stop

c.....initialize before numerical sparse assembling
      do 96 j=1,ndof
96    ip(j)=0
cccc blanked out below don't need dynamics
cccc        if (ianal .eq. 1) dm(j)=0.
cccc 96    ad(j)=0.
c 96    continue
c
cccc something not write with ncoef1???
cc                ncoef1=ncoef2

c                  write(6,*)'ncoef1 start assembly=',ncoef1
c     print *,'ncoef1=',ncoef1
c                 stop
cccc       do 95 j=1,ncoef1
cccc       if (ianal .eq. 1  .and. lumpmass .eq. 0) then
cccc       am(j)=0.
cccc       endif
cccc 95    an(j)=0.
c                 print *,'neltype=',neltype
c                 stop

c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 print *,'neltype=',neltype
c                 stop
c                 endif

c                  write(6,*),'neltype in assembly=',neltype
c                 stop

670               Continue !skipped above when getsolve=1
c**** START OF DO 98 ******************************
      DO 98 k=1,neltype

                  If(getsolve .eq. 1)Then
                  Goto 671
                  Endif

      numel=nel(k)
c                                 print *,'numel=',numel
c                 pause
c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 print *,'ianal=',ianal
c                 print *,'numel=',numel
c                 stop
c                 endif

c                   stop
      if (numel .eq. 0) go to 98
c.....loop for all elements
c                 print *,'start loop for all elements'
c                       print *,'ncoef1=',ncoef1
c                    print *,'numel =',numel
c                 pause
c                 stop
c                 print *,'Before Do 4'
c                 print *,'ux(1)=',ux(1)
c                 print *,'uy(1)=',uy(1)
c                 print *,'uz(1)=',uz(1)
c                 print *,'ux(2)=',ux(2)
c                 print *,'uy(2)=',uy(2)
c                 print *,'uz(2)=',uz(2)
c                 pause

      DO 4 iel=1,numel
c                 print *,'iel = ',iel
c                 print *,'********************'

c                 If(getforce .eq. 1)then
c                 goto 400
c                 endif

c                 pause
c                 print *,'numel=',numel
c                 pause
c                 stop
c     write(6,*) 'inside assembly: truss el #,nel(1) = ',iel,nel(1)
c.....get the global dof associated with each element
c                 print *,'ndofpe(k)=',ndofpe(k)
      iend=iel*ndofpe(k)
      istart=iend-ndofpe(k)
c                 print *,'iend=',iend
c                 print *,'istart=',istart
c                 print *,'ndofpe(k)=',ndofpe(k)
c                 pause

      do 76 j=1,ndofpe(k)
c                 print *,'j=',j
c                 print *,'je(istart+j)=',je(istart+j)
c                  write(6,*)'lm(j)=',lm(j)
      lm(j)=je(istart+j)
 76               continue
c 76              print *,'je(istart+j)=',je(istart+j)
c                 print *,'lm 2 inside assembly =',(lm(i),i=1,6)
c     pause

c                 print *,'iarc=',iarc
c                 pause
c                 stop

                  if(iarc .eq. 1)then
                  ix1dof=lm(1)
                  iy1dof=lm(2)
                  iz1dof=lm(3)
                  ix2dof=lm(4)
                  iy2dof=lm(5)
                  iz2dof=lm(6)
c                 print *,'lm(6)=',lm(6)
c     ux(1)=ux(1)+disp(ix1dof)
c                 uy(1)=uy(1)+disp(iy1dof)
c                 uz(1)=uz(1)+disp(iz1dof)
c                 ux(2)=ux(2)+disp(ix2dof)
c                 uy(2)=uy(2)+disp(iy2dof)
c                 uz(2)=uz(2)+disp(iz2dof)
c     ux(1)=ux(1)+qk(ix1dof)
c                 uy(1)=uy(1)+qk(iy1dof)
c                 uz(1)=uz(1)+qk(iz1dof)
c                 ux(2)=ux(2)+qk(ix2dof)
c                 uy(2)=uy(2)+qk(iy2dof)
c                 uz(2)=uz(2)+qk(iz2dof)
c                 print *,'qk(ix1dof)=',qk(ix1dof)
c                 pause
c                 stop
      ux(1)=qk(ix1dof)
                  uy(1)=qk(iy1dof)
                  uz(1)=qk(iz1dof)
                  ux(2)=qk(ix2dof)
                  uy(2)=qk(iy2dof)
                  uz(2)=qk(iz2dof)
c                 print *,'ix1dof=',ix1dof
c     print *,'iy1dof=',iy1dof
c     print *,'iz1dof=',iz1dof
c     print *,'ix2dof=',ix2dof
c     print *,'iy2dof=',iy2dof
c     print *,'iz2dof=',iz2dof
c                 print *,'checking ux uy uz'
c     print *,'ux(1)=',ux(1)
c     print *,'uy(1)=',uy(1)
c     print *,'uz(1)=',uz(1)
c     print *,'ux(2)=',ux(2)
c     print *,'uy(2)=',uy(2)
c     print *,'uz(2)=',uz(2)
c                 pause
c                 stop

                  If(getforce .eq. 1)then
c                 print *,'ix1dof=',ix1dof
c     print *,'iy1dof=',iy1dof
c     print *,'iz1dof=',iz1dof
c     print *,'ix2dof=',ix2dof
c     print *,'iy2dof=',iy2dof
c     print *,'iz1dof=',iz1dof
c     print *,'ux(1)=',ux(1)
c     print *,'uy(1)=',uy(1)
c     print *,'uz(1)=',uz(1)
c     print *,'ux(2)=',ux(2)
c     print *,'uy(2)=',uy(2)
c     print *,'uz(2)=',uz(2)
c                 write(6,*),'ix1dof=',ix1dof
c     write(6,*),'iy1dof=',iy1dof
c     write(6,*),'iz1dof=',iz1dof
c     write(6,*),'ix2dof=',ix2dof
c     write(6,*),'iy2dof=',iy2dof
c     write(6,*),'iz2dof=',iz2dof
c     write(6,*),'ux(1)=',ux(1)
c     write(6,*),'uy(1)=',uy(1)
c     write(6,*),'uz(1)=',uz(1)
c     write(6,*),'ux(2)=',ux(2)
c     write(6,*),'uy(2)=',uy(2)
c     write(6,*),'uz(2)=',uz(2)
c                 stop
                  endif

                  endif

c                 print *,'chk 2 inside assembly'
c                 stop
c                 pause

c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 print *,'ianal=',ianal
c                 stop
c                 endif

c
c                 print *,'chk before info_elstif inside assembly'
c                 pause
c                 print *,'lm 3 inside assembly =',(lm(i),i=1,6)

c                 print *,'xx inside assembly before info_= ',(xx(i),i=1,42)
c                 print *,'yy inside assembly before info_= ',(yy(i),i=1,42)
c                 print *,'zz inside assembly before info_= ',(zz(i),i=1,42)
c                 pause
c                 print *,'**** **** **** **** **** ****'
c                 print *,'before "call info_elstif" '
c                 pause

c                 print *,'xlength before info_elstif=',xlength
c                 print *,'xli before info_elstif=',xli
c                 pause
c                 print *,'je before info_elstif=',(je(ic),ic=1,10)
c                 pause
c                 print *,'B(_) before info_elstif in assembly=',(b(ic),ic=1,ndof)
c                 pause
      call info_elstif(iel,xlength,xl,cx,cy,cz,xx,yy,zz,ie,je
     $,propmat,propsect,youngm,area,rho,
     $ deltaxi,deltayi,deltazi,xli,
     $ xi,xj,yi,yj,zi,zj)

c                 print *,'xx inside assembly after info_= ',(xx(i),i=1,42)
c                 print *,'yy inside assembly after info_= ',(yy(i),i=1,42)
c                 print *,'zz inside assembly after info_= ',(zz(i),i=1,42)
```

```
c                 pause
c                 print *,'**** **** **** **** **** ****'
c                 print *,'after "call info_elstif" '
c                 print *,'B(_) after info_elstif in assembly=',(b(ic),ic=1,ndof)
c                 pause

c                 print *,'xlength after into_elstif=',xlength
c                 print *,'xli after info_elstif=',xli
c                 pause

      ndofped=ndofpe(k)
c     write(6,*) 'inside assembly: ndofped = ',ndofped
c                 print *,'chk before elstif inside assembly'
c                 stop
c                 print *,'lm 3a inside assembly =',(lm(i),i=1,6)

c                 pause
c     call elstif(ndofped,elk,be,
c    $            area,youngm,xlength,cx,cy,cz,maxdofpe,
c    $            ityp,xl,xli,dcltaxi,deltayi,deltazi,
c    $            finx,finy,finz,iarc)
c     call elstif(ndofpe,esm,be,

c                 print *,'xx inside assembly before elstif= ',(xx(i),i=1,42)
c                 print *,'yy inside assembly before elstif= ',(yy(i),i=1,42)
c                 print *,'zz inside assembly before elstif= ',(zz(i),i=1,42)
c                 print *,'++++++++++++++++++++++++++++++++++++++++++'
c                 print *,'Before elstif inside assembly'
c                 pause
c                 print *,'xlength before elstif=',xlength
c                 print *,'xli before elstif=',xli
c                 pause

c                 print *,'iel before call elstif=',iel
c                 pause
c                 print *,'iarea(_)=',(iarea(i),i=1,iel)
c                 pause
      call elstif(ndofpe,elk,be,propsect,iarea,
     $            area,youngm,xlength,cx,cy,cz,maxdofpe,
     $            xl,xli,deltaxi,dcltayi,deltazi,xx,yy,zz,
     $            iarc,xi,xj,yi,yj,zi,zj,ityp,lm,istep,iel,icheck,
     $            strainel,strainE,strainD,weight,Anew,Apart,Fin1,
     $            VED,VEDold,xlie,SED)

c                 print *,'xlength after elstif=',xlength
c                 print *,'xli after elstif=',xli
c                 pause

c                 print *,'xx inside assembly after elstif= ',(xx(i),i=1,4)
c                 print *,'yy inside assembly after elstif= ',(yy(i),i=1,4)
c                 print *,'zz inside assembly after elstif= ',(zz(i),i=1,4)
c                 print *,'++++++++++++++++++++++++++++++++++++++++++'
c                 print *,'After elstif inside assembly'
c                 pause

c                 print *,'lm 3b insidc assembly =',(lm(i),i=1,6)
c                 pause
c                  print *,'Chock after elstif'

c                 print *,'idir after elstif in assembly=',(idir(ic),ic=1,ndof)
c                 pause

c                 DO 919 i=1,6
c                 DO 919 j=1,6
c919              print *,'elk after elstif = ',elk(i,j)
c                 pause
c     stop
c                 print *,'getforce=',getforce
c                 print *,'finx(1)=',finx(1)
c                 print *,'finy(1)=',finy(1)
c                 print *,'finz(1)=',finz(1)
c                 print *,'finx(2)=',finx(2)
c                 print *,'finy(2)=',finy(2)
c                 print *,'finz(2)=',finz(2)
c                 pause
c                 stop

                  If(getforce .eq. 1)then
                    b(ix1dof)=finx(1)+b(ix1dof)
                    b(iy1dof)=finy(1)+b(iy1dof)
                    b(iz1dof)=finz(1)+b(iz1dof)
                    b(ix2dof)=finx(2)+b(ix2dof)
                    b(iy2dof)=finy(2)+b(iy2dof)
                    b(iz2dof)=finz(2)+b(iz2dof)

c                 write(6,*)'b x&y=',(b(i),i=1,ndof)
c                 print *,'b x,y,z=',(b(i),i=1,ndof)
c                 pause
c                 stop
c                 goto 4
      endif

c                 print *,'lm 4 inside assembly =',(lm(i),i=1,6)
c                 pause
c                 if(it .eq. 2)then
c                 print *,'it =2'
c                 print *,'ianal=',ianal
c                 stop
c                 endif
c                 print *,'ianal =',ianal

      if (ianal .eq. 1) then
      call elmass(rho,area,xlength,lumpmass,elm,ndofped,maxdofpe,
     $cx,cy)
      endif
c
c     write(6,*) '*** passed elstiff ***'
c                  print *,'*** passed elstiff ***'
c                 print *,'ndofpe(k)=',ndofpe(k)
c                 pause
c                 stop

c                 DO 920 i=1,6
c                 DO 920 j=1,6
c920              print *,'elk after elstif = ',elk(i,j)

c                 print *,'zz 1 inside assembly after elstif= ',(zz(ic),ic=1,3)
c                 pause


c677   Continue !skip to here from start if getsolve=1, just solving

c.....convert element [k] & [m] from 2-D array into 1-D array (column-wise)
      do 78  j=1,ndofpe(k)
c                 print *,'lm 5 inside assembly =',(lm(icc),icc=1,6)

c                 print *,'zz 1a inside assembly after elstif= ',(zz(ic),ic=1,3)
c                 pause
      do 77 jj=1,ndofpe(k)
c                 print *,'lm 6 inside assembly =',(lm(ic),ic=1,6)
      locate=jj+(j-1)*ndofpe(k)
c                 print *,'lm 6a inside assembly =',(lm(ic),ic=1,6)

c                 print *,'zz 1b inside assembly after elstif= ',(zz(ic),ic=1,3)
c                 pause
cc                print *,'ae(locate)=',ae(locate)

ccc Something wrong with lm ??????????????????????????
c                 do 812 im=1,6
c812              lm2(im)=lm(im)

c                 print *,'zz 1c inside assembly after elstif= ',(zz(id),id=1,3)
c                 pause
c                 print *,'ij, j, elk(ij,j)=',ij,'  ',j,'  ',elk(ij,j)
c                 pause
c                  print *,'locate, ae(locate) before=',locate,'  ',ae(locate)
c                 pause
      ae(locate)=elk(jj,j)
c                  print *,'locate, ae(locate) after=',locate,'  ',ae(locate)
c                 pause
c                 print *,'zz 1d inside assembly after ae(_)= ',(zz(ic),ic=1,3)

c                 Do 922 id=1,3
c                 zz(id)=0.0 !Force zz to zero, something wrong above!!!
c922   continue
c                 print *,'zz 1d forcing to zero= ',(zz(ic),ic=1,3)

c                  pause

c                 do 813 im=1,6
c813   lm(im)=lm2(im)
                        !  for [k]
cc                print *,'lm 6b insidc assembly =',(lm(ic),ic=1,6)
c                  write(6,*)'locate=',locate
c                  write(6,*)'ae(locate)=',ae(locate)
c       print *,'locate end assembly=',locate
c                    print *,'ae(locate) end assembly=',ae(locate)
c                 pausc
c                   print *,'ij, j = ',ij,'  ',j
c                   print *,'elk(ij,j)=',elk(ij,j)
c                 pausc
c                  write(6,*)            'jj, j, locate         = ',ij,'  ',j,' ',locate
c                  write(6,*)'elk(ij,j)=',elk(ij,j)
c                    pause
      if (ianal .eq. 1) then
      tempo1(locate)=elm(jj,j)    !  for [m]
c                               write(6,*)'tempo1=',tempo1
      endif
cc                print *,'lm 7 inside assembly =',(lm(ic),ic=1,6)
c                 pausc
c                 print *,'zz 2 inside assembly lm 7= ',(zz(icc),icc=1,3)
c                 pause
 77   continue
 78   continue

c                 print *,'ae(_)=',(ae(ic),ic=1,20)
c                 pause
c
c                 print *,'lm 8 insidc assembly =',(lm(ic),ic=1,6)
c                   stop
c                  print *,'going to numass'
c                  print *,'ncoef1=',ncoef1
c                  print *,'ia before numass in assembly=',(ia(ic),ic=1,10+1)
c                  pause
c not correct...something wrong with ncoef1, again
c                  ncoef2=ncoef1
c                  stop
c.....sparse "numerical" assembly
c                   print *,'ia1 before numass in assembly=',(ia1(i),i=1,ndof+1)
c                   print *,'ia2 before numass in assembly=',(ia2(i),i=1,ndof+1)
c                  write(6,*)'ia2 before numass in assembly=',(ia2(i),i=1,ndof+1)
c                  stop
c                  print *,'ja(1) before numass in assembly=',ja(1)
c                  pause
c                  print *,'lm inside assembly before numass =',(lm(i),i=1,6)

c                  print *,'iboundc before numass in ably=',(iboundc(ic),ic=1,ndof)
c                  pause
c...iboundc=idir in subroutine numass
c...tempo1=elmass in subroutine numass

c                  print *,'xx inside assembly before numass= ',(xx(i),i=1,42)
c                  print *,'yy inside assembly before numass= ',(yy(i),i=1,42)
c                  print *,'zz inside assembly before numass= ',(zz(i),i=1,42)
c                  print *,'+++++++++++++++++++++++++++++++++++++++++++'
c                  print *,'+++++++++++++++++++++++++++++++++++++++++++'
c                  print *,'Before numass inside assembly'
c                  pause

c     print *, ' b(-) before numass in assembly= ',( b(i),i=1,ndof)
c                  pause
c                  print *,'getforce=',getforce
c                  pause
```

```
                     If(getforce .eq. 1)Then
c                    print *,'getforce = 1, skip numass'
                     goto 405
                     Endif

671  Continue !skip to here from start if getsolve=1, just solving

c    print *, '*** before numass ***'
c    print *,'ad(-) = ',(ad(i),i=1,ndof)
c    print *, 'an(-) = ',(an(i),i=1,ncoef1)
c    print *, ' b(-) before numass in assembly= ',( b(i),i=1,ndof)
c                    print *,'iboundc before numass=',(iboundc(ic),ic=1,ndof)
c                    pause

      call numass(ia,ja,iboundc,ae,be,lm,ndofped,an,ad,b,ip,
     $tempo1,dm,am,ianal,lumpmass,ndof,ncoef1,inc,istep)
c    print *, ' b(-) after numass in assembly= ',( b(i),i=1,ndof)
c                    pause
405  continue !dummy line

c                    print *,'xx inside assembly after numass= ',(xx(i),i=1,42)
c                    print *,'yy inside assembly after numass= ',(yy(i),i=1,42)
c                    print *,'zz inside assembly after numass= ',(zz(i),i=1,42)
c                    print *,'++++++++++++++++++++++++++++++++++++++++'
c                    print *,'++++++++++++++++++++++++++++++++++++++++'
c                    print *,'After numass inside assembly'
c                    pause

c     call numass(ia2,ja,iboundc,ae,be,lm,ndofped,an,ad,b,ip,
c    $tempo1,dm,am,ianal,lumpmass,ndof,ncoef1)
c    write(6,*) '*** passed numass ***'
c    write(6,*) 'ad(-) = ',(ad(i),i=1,ndof)
c    write(6,*) 'an(-) = ',(an(i),i=1,ncoef1)
c    write(6,*) ' b(-) = ',( b(i),i=1,ndof)
c    print *, '*** passed numass ***'
cc    print *,'ad(-) = ',(ad(i),i=1,ndof)
cc    print *, 'an(-) = ',(an(i),i=1,ncoef1)
c    print *, ' b(-) after numass in assembly= ',( b(i),i=1,ndof)
c                    pause
c                     print *,'ncoef1 after numass=',ncoef1
c                    stop

c                     print *,'ia2 after numass in assembly=',(ia2(i),i=1,ndof+1)
c                    stop
c                     ncoef1=ncoef2
c                     print *,'ncoef1=',ucoef1
c     print *, 'ad diag stiff = ',(ad(i),i=1,ndof)
c     print *, 'an off diag stiff = ',(an(i),i=1,ncoef1)
c                    stop
c    write(6,*) '*** passed numass ***'
c                    write(6,*) 'iel =',iel
c    write(6,*) 'ad(-) = ',(ad(i),i=1,ndof)
c    write(6,*) 'an(-) = ',(an(i),i=1,ncoef1)
c    write(6,*) ' b(-) = ',( b(i),i=1,ndof)
c    print *, '*** passed numass ***'
c    print *, 'ad(-) = ',(ad(i),i=1,ndof)
c    print *, 'an(-) = ',(an(i),i=1,ncoef1)
c    print *, ' b(-) = ',( b(i),i=1,ndof)

c                    print *,'xx inside assembly end do 4= ',(xx(i),i=1,42)
c                    print *,'yy inside assembly end do 4= ',(yy(i),i=1,42)
c                    print *,'zz inside assembly end do 4= ',(zz(i),i=1,42)
c                    print *,'&&&&&&&&&&& end do 4 &&&&&&&&&&&&&&'
c                    pause

chk                  If(icheck .eq. 1) then

chk     print *, 'ad diag stiff final = ',(ad(i),i=1,ndof)
chk     print *, 'an off diag stiff final = ',(an(i),i=1,ncoef1)

chk                   pause
chk                  Endif

 4   continue

 98  continue
c                    stop
c     print *, 'ad(-) = ',(ad(i),i=1,ndof)

c                    stop
c     write(6,*), 'ad diag stiff final = ',(ad(i),i=1,ndof)
c     write(6,*), 'an off diag stiff final = ',(an(i),i=1,ncoef1)
c     write(6,*),' b(-) at end assembly = ',( b(i),i=1,ndof)
c     print *, 'ad diag stiff final = ',(ad(i),i=1,ndof)
c                     pause

cc     print *, 'an off diag stiff final = ',(an(i),i=1,ncoef1)
c
c                    stop
      return
      end

c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      subroutine generalip(nel,ndofpe,neltype,maxdofpe,nboundc
    $,           ndofpn,numnodes,loadof,nummat,nlayers,nsect
    $,           lumpmass,ndof,ieall,jeall,ianal,neig,iarc
    $,           itype,numlim)

c
c if "iarc"=1 then use Arc Length method
c if "itype"=6 then 2d nonlinear truss element
c
c......Purpose: Input some general input information
c
      dimension nel(1),ndofpe(1)
c      common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)
c
                     print *,'check1'
      read(5,*) nboundc,ndofpn,numnodes,loadof,nummat
     $,nlayers,nsect,lumpmass,ianal,neig,iarc,itype,itruss,numlim
c
```

```
c......ndofpe(1,2,3,4,5,6)=(4,12,24,15,99,99)
c                  = 2-d truss
c                  = 3-d beam
c                  = 2-d BFS rectangular plate (bending+membrane)
c                  = 3-d MIN3 triangular plate
c                  =    "future" element type
c                  =    "future" element type
                   print *,'check2'
      read(5,*) (nel(i),i=1,neltype)
      read(5,*) (ndofpe(i),i=1,neltype)
c      write(6,*) 'nboundc,ndofpn,numnodes,loadof'
c      write(6,*) 'nummat,nlayers,nsect,lumpmass,ianal,neig'
c      write(6,*)  nboundc,ndofpn,numnodes,loadof
c      write(6,*)  nummat,nlayers,nsect,lumpmass,ianal,neig
c      write(6,*) 'nel(neltype) = ',(nel(i),i=1,neltype)
c      write(6,*) 'ndofpe(neltype) = ',(ndofpe(i),i=1,neltype)
      ndof=numnodes*ndofpn
c                  print *,'ndofpe in gen = ',ndofpe
c                  print *,'ndof in gen = ',ndof
                   print *,'nel inside generalip=',nel
c                  pause
c                  stop
c
       maxdofpe=6     ! for 3-d truss
        icall=0
        jeall=0
       do 1 i=1,neltype
       if (nel(i) .eq. 0) go to 1
        if (ndofpe(i) .gt. maxdofpe) maxdofpe=ndofpe(i)
        ieall=icall+nel(i)
        jeall=jcall+nel(i)*ndofpe(i)
 1      continue
                    print *,'end of generalip'
c
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      subroutine supernode(ndof,isupn)
      dimension isupn(1)
c
c     Purposes: Find super node information (for unrolling purpose)
c
      do 11 i=1,ndof
      isupn(i)=1
 11   continue
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      subroutine minfills(ndof,iperm)
      dimension iperm(1)
c
c     Purpose: Reordering algorithms, to minimize fill-in terms
c
      do 1 i=1,ndof
      iperm(i)=i
 1    continue
c                   print *,'iperm inside minfills=',(iperm(i),i=1,ndof)
c                   stop
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine copy_int(n,iarray,icopy)
      dimension iarray(1),icopy(1)
c                   print *,'Check inside copy_int'
c                   print *,'n=',n
c                   print *,'iarray(1)=',iarray(1)
       do 2 i=1,n
       icopy(i)=iarray(i)
c                   print *,'icopy(i) inside copy_int=',icopy(i)
 2     continue
      return
      end
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
c     A R C   L E N G T H
c
c This subroutine added for Arc Length method.  Also
c updated elstif for 2D nonlinear truss.
c Rev 3/16/04
c
C**********************************************************************
                   Subroutine rikarc(ie,jc,lm,neltype,nel,ndofpe,ndofpn,
     $ numnodes,x,y,z,dqtot,
     $ ndof,b,loadof,
     $ iboundc,nboundc,ia,
c     $ iboundc,nboundc,ia1,ia2,
     $ icall,ji,iet,jet,
     $ itempo1,exfor,
     $ ja,
     $ ip,ad,an,propmat,propsect,be, !call assembly
c     $ ip,ad,an,propmat,propsect,elk,be, !call assembly
     $ ae,maxdofpe,
     $ ianal,lumpmass,dm,am,elm,tempo1,
     $ iperm,  !call minfills
     $ in,ncoef2,ju, !call symfactd
     $ iu,iut,  !call transad
     $ isupn, !call supernode
     $ di,un,iup,iopf,  !call numfa1d
     $ iopb,  !call fbed
     $ disp,iarc,jeall,ux,uy,
     $ qk,iarca,
     $ strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $ Fin2,VED,VEDold,xlic,exlkdl0,exlkdli,exlkdlk,SED,iexlk,
     $ weightotal,Nevenodd,N)
c     $ ja,qk)
c     $ ja,esm,elk,qk)
C**********************************************************************
C***           PROGRAM: RIK2DOF
C***
```

```
C***   Program to trace the equilibrium path for a geometrically
C***   nonlinear single degree of freedom truss.
C***
C***   Input Variables:
C***   dl0 = starting incremental external load multiplier,
C**       delta lambda 0, (real, scalar)
C***   dlmax = maximum allowable incremental external load
C***         multiplier, delta lambda max, (real, scalar)
C***   dlmin = minimum allowable incremental external load
C***         multiplier, delta lambda min, (real, scalar)
C***   exfor = external force, Q, (real, vector)
C***   exlmx = maximum allowable total external load
C***         multiplier, lambda max, (real, scalar)
C***   itedes = number of desired iterations for
C***          convergence, (integer, scalar)
C***   itemax = maximum allowable number of iterations,
C***          (integer, scalar)
C***   mxstp = maximum allowable number of load steps,
C***         (integer, scalar)
C***   toldis = convergence tolerance for displacements,
C***         (real, scalar)
C***   tolfor = convergence tolerance for residual forces,
C***         (real, scalar)
C***
C***   Output Variables:
C***   exlk = total external load multiplier at step k (or i),
C***        lambda k (or lambda i), (real, scalar)
C***   qk = total displacements at step k (or i),
C***      qk (or qi), (real, vector)
C***
C***   Intermediate Variables:
C***   dli = incremental external load multiplier at step i,
C***       delta lambda i, (real, scalar)
C***   dlim1 = incremental external load multiplier at step i-1,
C***         delta lambda i-1, (real, scalar)
C***   dlk = incremental external load multiplier at step k,
C***       delta lambda k, (real, scalar)
C***   dqi = incremental displacements at step i, delta qi,
C***       (real, vector)
C***   dqim1 = incremental displacements at step i-1, delta qi-1,
C***         (real, vector)
C***   dqk = incremental displacements at step k, delta qk,
C***       (real, vector)
C***   dqk1 = part 1 of delta qk, delta qkI, (real, vector)
C***   dqk2 = part 2 of delta qk, delta qkII, (real, vector)
C***   dqtot = total displacements at load step i, delta qtot,
C***         (real, vector)
C***   dqsq = storage for dqtot*dqtot, (real, scalar)
C***   ds0 = magnitude of initial tangent vector, delta s0,
C***       (real, scalar)
C***   dsi = magnitude of the tangent vector at load step i,
C***       delta si, (real, scalar)
C***   dsim1 = magnitude of the tangent vector at load step i-1,
C***         delta si-1, (real scalar)
C***   dsmax = maximum allowable magnitude of the tangent
C***         vector, delta smax, (real, scalar)
C***   dsmin = minimum allowable magnitude of the tangent
C***         vector, delta smin, (real, scalar)
C***   fintk = internal force at load step k (or i),
C***         Fk (or Fi), (real, vector)
C***   iconv = convergence flag, iconv=0 means toldis and
C***         and tolfor have been met, (integer, scalar)
C***   iflag = error flag for iteration on the normal
C***         plane, iflag .ne. 0 means maximum iterations
C***         have been exceeded, (integer, scalar)
C***   istep = load step counter (integer, scalar)
C***   it = iteration counter (integer, scalar)
C***   resk = residual force at load step k (or i),
C***        FRk (or FRi), (real, vector)
C***   scali = scale factor for load step i, Si, (real, vector)
C***   stifi = tangent stiffness at i, KTi, (real, matrix)
C***   stifk = tangent stiffness at k, KTk, (real, matrix)
C***
C***
C*********************************************************************
*
      implicit real*8(a-h,o-z)
c     parameter (neq=2)

      common/RIK/ stifi(60,60),dqk1(5000),
     $    dqk2(5000),fintk(5000),
     $    resk(5000),scali(5000),work1(5000),indx(5000),neq

c     common/deltaqi/dqi(500)

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
c     common/disp/ux(2),uy(2),uz(2),disp(500)
      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/testing/toldis,tolfor,exlk,iconv
      common/nodeforc/ibstor(500),bvstor(500)

                    common/jam/ja2(2000)

c     common/eletype/ityp,itruss  !ityp = 6 for 2-D, =12(3D truss)

      double precision, allocatable:: dqi(:),dqim1(:)
      integer, allocatable:: itempo2(:)

                    dimension dqtot(*)

c     double precision, allocatable:: ja(:)

                    dimension disp(*),b(*),ia(*),iperm(*),qk(*)
c                   dimension disp(*),b(*),ia1(*),ia2(*),iperm(*)
                    dimension ie(*),je(*),ict(*),jet(*),ac(*),ad(*)
                    dimension iboundc(*),an(*),ux(*),uy(*),di(*),exfor(*)
                    dimension x(*),y(*),z(*),ip(*),lm(1),iu(*),iarea(*)
c                   dimension ja(*),x(*),y(*),z(*),ip(*),lm(1),iu(*)

                    dimension ja(*)

                    dimension strainel(*),strainE(*),strainD(*),weight(*)
                    dimension Anew(*),Fin1(*),Fin2(*),VED(*),VEDold(*)

c                   If(iopt .eq. 1)goto 2 !skip, going thru rikarc again in optimizier
c     double precision, allocatable:: dqtot(:),dqi(:),dqim1(:)
c     integer, allocatable:: itempo2(:)
c2                  continue

c                   dimension elk(maxdofpe,maxdofpe)

c ... open files for output
c     open(15,file='fearc_out.txt',status='old')
c     open(12,file='fearc_out_F.txt',status='old')
c     open(13,file='fearc_out_q1.txt',status='old')

c     open(15,file='fea3d2_out.txt',status='old')
c     open(12,file='fea3d2_out_F.txt',status='old')
c     open(13,file='fea3d2_out_q1.txt',status='old')


c     open(14,file='fea3d2_out_q2.txt',status='old')

c                   allocate (ja(50))

c                   print *,'Start Rik'

                    neq=ndof
c                   print *,'numnodes=',numnodes
c                   print *,'x start rik =',(x(i),i=1,numnodes)
c                   print *,'y start rik =',(y(i),i=1,numnodes)
c                   print *,'z start rik =',(z(i),i=1,numnodes)
c                   stop
c                   print *,'b start rikarc = ',(b(i),i=1,ndof)
c                   stop
c     print *,'jgsm = ',jgsm
cc     print *,'np = ',np
c     print *,'neq inside Rikarc = ',neq
c     print *,'a at start of RIK= ',a
c     print *,'dqi at start of RIK=',dqi
c      print *,'start of RIK'
c                    print *,'ityp start RIK=',ityp
c                    print *,'ndof start RIK=',ndof
c                   stop
c                   print *,'maxdofpe=',maxdofpe
c      stop
c c     pause
c ... define user input
c** dl0 = starting incremental external load multiplier
c     dl0 = 2.5d-2
ccccc      dl0 = 1.d-2
c      dl0 = 1.d-1
c4       dl0 = 1.d0
cc      dl0 = 1.d-3
ccc       dl0 = .001
C***      dlmax = maximum allowable incremental external load
C***            multiplier, delta lambda max, (real, scalar)
C***      dlmin = minimum allowable incremental external load
C***            multiplier, delta lambda min, (real, scalar)
c     dlmax = 2.6d-2
ccccc       dlmax = 1.d-2
ccccc       dlmin = 1.d-3
ccc       dlmax = 1.d-3
ccc       dlmin = 1.d-4
cc       dlmax = 1.d-4
cc       dlmin = 1.d-5
ccccccc         dlmax = .1
ccccccc         dlmin = .01
cccccc         dlmax = 1.
cccccc         dlmin = .1
c4        dlmax = 1000.
c4        dlmin = 10.
cxx         dlmin = 1.
c** Use below for truss1
c       dlmax = 1.d-0
c       dlmin = 1.d-1
c** Use below for truss2
c       dlmax = 1.d+1
c       dlmin = 1.d-0
c** read in the intial forces from big "a"
cc      print *,'jgf = ',jgf
c     stop
cc      print *,'exfor at start of RikArc = ',exfor
c     stop
c     exfor(1) = 1.55d0
c     exfor(2) = 1.45d0
C***      exlmx = maximum allowable total external load
C***            multiplier, lambda max, (real, scalar)
ccccc      exlmx = 1.d0
cccc       exlmx = .5d0
cc        exlmx = .2d0
cc      exlmx = .1d0
c4       exlmx = 100.d0

c     itedes = 4
c      itedes = 1
c     itemax = 20000
c     itemax =999999
c     itemax = 1500
c     itemax = 900000
      itemax = 100000
cc      mxstp = 60
czzz      mxstp = 1500
ccc       mxstp = 25000
       mxstp = 1000000 !istep
cccccc      toldis = 1.d-3
cccccc      tolfor = 1.d-3
cxx      toldis = 1.d-2
cxx      tolfor = 1.d-2
cccc      toldis = .5
cccc      tolfor = .5
c4      toldis = 1.d-1
c4      tolfor = 1.d-1
```

```
c       toldis = 1.5d0
c       tolfor = 1.5d0
C*********************************************************
c** Truss_2bay_3D.dat
C** Symmetric
C** This is 6 nodes, 11 element
c       itedes = 1
c        dl0 = .01d0
c        dlmax = .001d0
c         dlmin = .0001d0
c       exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*********************************************************
C*********************************************************
c** Two Bar Truss
C** Symmetric
C** This is 3 nodes, 2 element
cc       itedes = 1
cc        dl0 = .01d0
cc        dlmax = .01d0
cc         dlmin = .001d0
cc       exlmx = 1.d0
cc      toldis = 1.d-6
cc      tolfor = 1.d-6
c                       print *,'Check After input'
c                       pause
C*********************************************************
C*********************************************************
c** Tripod
C** Crisfield 3.10.2
C** This is 4 nodes, 2 element
cpod       itedes = 1
cpod        dl0 = .1d0
cpod        dlmax = .001d0
cpod         dlmin = .0001d0
cpod       exlmx = 1.d0
cpod       toldis = 1.d-6
cpod       tolfor = 1.d-6
c      toldis = 1.d-12
c      tolfor = 1.d-12
C*********************************************************
C*********************************************************
C** Crisfield truss, ex 22.4a, vol2 chap 22.4, page 407
C** cris_22_4a_in.dat  6nodes, 9 elements
c       itedes=1
c       dl0 = .1d0
c       dlmax = .01d0
c        dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*********************************************************
C*********************************************************
C** Duc truss, truss11_3d_in.dat
C** Duc truss, truss11_3d_xz_in.dat
C** This is a 6 nodes, 11 elements,
c       itedes=1
c       dl0 = .01d0
c       dlmax = .0001d0
c        dlmin = .00001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*********************************************************

C*********************************************************
C** Duc truss, truss10_3d_in.dat
C** This is a 6 nodes, 10 elements,
c       itedes=1
c       dl0 = .01d0
c       dlmax = .001d0
c        dlmin = .0001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*********************************************************
C** Duc truss, truss7_3d_in.dat
C** This is a 5 nodes, 7 elements,
ctruss       itedes=1
ctruss       dl0 = .01d0
ctruss       dlmax = .001d0
ctruss        dlmin = .0001d0
ctruss      exlmx = 1.d0
ctruss     toldis = 1.d-4
ctruss     tolfor = 1.d-4
C*********************************************************
C** The following are used for test problelm Truss3in.dat& T4in.dat
C** This is a 2 ele, 3 node problem, results agree with Crisfield
c       dl0 = .1d0
c        dlmax = .001
c        dlmin = .0001
c       exlmx = 1.d0
c      toldis = .00001
c      tolfor = .00001d0
C*********************************************************
C** The following are used for test problelm T3in.dat
C** This is a 3 ele, 3 node problem, results agree with Crisfield
c       dl0 = .1d0
c        dlmax = .01
c        dlmin = .001
c       exlmx = 1.d0
c      toldis = .01d0
c      tolfor = .01d0
C*********************************************************
C** The following are used for test problelm T3xin.dat & T3ain.dat
C** This is a 6 ele, 4 node problem, results agree with Crisfield
c       dl0 = .1d0
c        dlmax = .01
c        dlmin = .001
c       exlmx = 1.d0
c      toldis = .00001d0
c      tolfor = .00001d0
C*********************************************************
C** The following are used for test problelm T3xin.dat
C** This is a 6 ele, 4 node problem, results agree with nastran
c       dl0 = .1d0
c        dlmax = .1
c         dlmin = .01
c       exlmx = 1.d0
c      toldis = .000001d0
c      tolfor = .01d0
C*********************************************************
C** The following are used for test problelm  T4xxxin.dat GOOD
C** This is a 10 ele, 7 node problem, results agree with nastran
c       dl0 = .1d0
c        dlmax = .1d0
c         dlmin = .01d0
c       exlmx = 1.d0
c      toldis = 1.d-9
c      tolfor = 1.d-9
C*********************************************************
C** Crisfield small arch,  arch20in.dat
C** This is a 10 nodes, 20 element,
c       dl0 = .1d0
c        dlmax = .001d0
c         dlmin = .0001d0
c       exlmx = 1.d0
c      toldis = 1.d-7
c      tolfor = 1.d-7
C*********************************************************
C** Crisfield small arch,  arch18in.dat
C** This is a 18 nodes, 41 element,
c       dl0 = .01d0
c        dlmax = .001d0
c         dlmin = .0001d0
c       exlmx = 1.d0
c      toldis = 1.d-5
c      tolfor = 1.d-5
C*********************************************************
C*********************************************************
C** Crisfield small arch,  arch10_3D_in.dat
C** This is a 10 nodes, 21 element,
c        itedes=1
c        dl0 = .001d0
c        dlmax = .001d0
c         dlmin = .0001d0
c       exlmx = 1.d0
c       toldis = 1.d-6
c       tolfor = 1.d-6
C*********************************************************
C** Crisfield small arch,  arch18_3D_in.dat
C** This is a 18 nodes, 41 element,
c        itedes=5
c        dl0 = .01d0
c        dlmax = .005d0
c         dlmin = .0005d0
c       exlmx = 1.d0
c       toldis = 1.d-6
c       tolfor = 1.d-6
C*********************************************************
C** Crisfield large symmetry arch,  arch22in.dat
C** This is a 22 nodes, 51 elements,
cc        dl0 = .1d0
cc        dlmax = .1d0
cc         dlmin = .01d0
cc       exlmx = 1.d0
cc      toldis = 1.d-3
cc      tolfor = 1.d-4
C*********************************************************
C** Crisfield small arch,  arch42_3D_in.dat
C**    arch3din.dat
C** This is a 42 nodes, 101 element,
c       itedes=1
c       dl0 = .1d0
c       dlmax = .01d0
c        dlmin = .0001d0
c      exlmx = 1.d0
c      toldis = 1.d-4
c      tolfor = 1.d-4
C*********************************************************
C** Crisfield small arch,  arch42in.dat
C**    arch10_3D_in.dat
C** This is a 21 nodes, 10 element,
c                       itedes=1
c       dl0 = .01d0
c       dlmax = .01d0
c        dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-4
c      tolfor = 1.d-4

carch                   itedes=1
carch                    dl0 = .1d0
carch                    dlmax = .1d0
carch                     dlmin = .001d0
carch                   exlmx = 1.d0
carch                  toldis = 1.d-6
carch                  tolfor = 1.d-6

c                       itedes=1
c       dl0 = .01d0
c       dlmax = .01d0
c        dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-4
c      tolfor = 1.d-4

C*********************************************************
C*********************************************************
C** Shallow_truss & Khot 4element
C**  Khot paper
C** This is a 13 nodes, 23 elements,
c                       itedes=1
```

```
c       dl0 = .01d0
c       dlmax = .01d0
c       dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-4
c      tolfor = 1.d-4

                        itedes=1
       dl0 = .1d0
       dlmax = .10d0
        dlmin = .001d0
      exlmx = 1.d0
     toldis = 1.d-4
     tolfor = 1.d-4

c                       itedes=1
c       dl0 = .01d0
c       dlmax = .01d0
c        dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-4
c      tolfor = 1.d-4

C*************************************************************



C** Duc frame example  frame_in.dat
C**
C** This is a 4 nodes, 5 element,
c                       itedes=1
c       dl0 = .01d0
c       dlmax = .001d0
c        dlmin = .0001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6

c                       itedes=2
c       dl0 = .025d0
c       dlmax = 8.d-3
c        dlmin = 5.d-3
c      exlmx = 1.d0
c      toldis = 1.d-3
c      tolfor = 1.d-4
C*************************************************************

C*************************************************************
C**    t3a_in.dat
C** This is a 3 nodes, 3 element,
ct3a        itedes=1
ct3a        dl0 = .00001d0
ct3a        dlmax = 1.d-3
ct3a         dlmin = 1.d-4
ct3a       exlmx = 1.d0
ct3a       toldis = 1.d-6
ct3a       tolfor = 1.d-6
C*************************************************************

C** 3D Truss , Td3_in.dat
C** This is a 4 nodes, 3 elements,
c       dl0 = .1d0
c       dlmax = .01d0
c        dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-5
c      tolfor = 1.d-5
C*************************************************************
C** 3D Truss , Dome_in.dat & Dome2_in.dat
C** This is a 13 nodes, 24 elements,
cc                      If (exlk .gt. .99 .and. exlk .lt. 1.0)then
cc                      print *,'exlk .gt. .99 at start rik=',exlk
cc       dl0 = .00001d0
cc       dlmax = .00001d0
cc        dlmin = .000001d0
cc                      pause
cc                      Else
cdome       itedes=1
cdome        dl0 = .1d0
cdome        dlmax = .1d0
cdome         dlmin = .01d0
cdome       exlmx = 1.d0
cdome       toldis = 1.d-4
cdome       tolfor = 1.d-4
C*************************************************************
C** Crisfield 2_2 Truss , Cris_sodf_in.dat
C** This is a 2 nodes, 1 element,
csdof       itedes=1
csdof        dl0 = .01d0
csdof        dlmax = .001d0
csdof        dlmin = .001d0
csdof       exlmx = 1.d0
csdof      toldis = 1.d-4
csdof      tolfor = 1.d-4

C*************************************************************
C*************************************************************
C** Khot 2 ele unsymmetric Truss , Khot_2ele_in.dat
C** This is a 3 nodes, 2 element,

ckhot2       itedes=1
ckhot2       dl0 = .01d0
ckhot2       dlmax = .01d0
ckhot2        dlmin = .001d0
ckhot2       exlmx = 1.d0
ckhot2       toldis = 1.d-4
ckhot2       tolfor = 1.d-4

cc       itedes=1
cc       dl0 = .0001d0
cc       dlmax = .0001d0
cc        dlmin = .00001d0
cc       exlmx = 1.d0
cc       toldis = 1.d-4
cc       tolfor = 1.d-4

C*************************************************************
C*************************************************************
C** snap_in.dat
C** This is a 4 nodes, 3 elements,
csnap       itedes=1
csnap        dl0 = .01d0
c       dl0 = .5d0
csnap        dlmax = .1d0
csnap         dlmin = .00001d0
csnap       exlmx = 1.d0
csnap       toldis = 1.d-4
csnap       tolfor = 1.d-4

C*************************************************************
C** 3D Truss , twobar_in.dat !Similar to Crisfield 22.1
C** large bifurcation
C** This is a 3 nodes, 2 elements,
ctwo       itedes=1
ctwo        dl0 = .001d0
c       dl0 = .5d0
ctwo        dlmax = .001d0
ctwo         dlmin = .0001d0
ctwo       exlmx = 1.d0
ctwo       toldis = 1.d-6
ctwo       tolfor = 1.d-6

C*************************************************************
C** The following are used for test problelm T4xin.dat & T4xxin.dat & T4in.dat
C** This is a 10 ele, 7 node problem, results agree with nastran
c       dl0 = .1d0
c       dlmax = .01d0
c       dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-6
c      tolfor = 1.d-6
C*************************************************************
C** The following are used for test problelm T5xxxin.dat & T4xxin.dat
C** This is a 10 ele, 7 node problem, results agree with nastran
c       dl0 = .1d0
c       dlmax = .01d0
c       dlmin = .001d0
c      exlmx = 1.d0
c      toldis = 1.d-5
c      tolfor = 1.d-4
C*************************************************************
C** The following are used for test problelm T7in.dat
C** This is a 4 ele, 5 node problem, results agree with Crisfield
c       dl0 = .01d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c      toldis = .01d0
c      tolfor = .01d0
C*************************************************************
C*************************************************************
C** The following are used for test problelm T7Ain.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
C** using these varibles, this takes a long time 30+ min
c       dl0 = .1d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c      toldis = .01d0
c      tolfor = .01d0
C** The following are used for test problelm T7Bin.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
C** try using differcnt varibles for faster
c       dl0 = 1.d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c      toldis = .01d0
c      tolfor = .01d0
C** The following are used for test problelm T7Cin.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
C** try using different varibles for faster
c       dl0 = 1.d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = .1d0
c      toldis = .01d0
c      tolfor = .01d0
C** The following are used for test problelm T7Din.dat
C** This is a 6 ele, 5 node, 3 support problem, results agree with Crisfield
c       dl0 = .01d0
c       dlmax = .01
c       dlmin = .001
c      exlmx = 1.d0
c      toldis = .001d0
c      tolfor = .001d0
C** The following are used for test problelm T7iEn.dat
C** This is same as T7D above but use 2 supports instead of 3
C** This is a 6 ele, 5 node, 2 support problem
C** Trying to get a solution, do I need right bracketing or is program wrong?
c       dl0 = .01d0
c       dlmax = .01d0
c       dlmin = .001d0
c      exlmx = 1.d0
c .9 works, try smaller
c      toldis = .9d0
c      tolfor = .9d0
C*************************************************************
C*************************************************************
C** The following are used for test problelm T11in.dat
C** This is a 4 ele, 5 node problem, results agree with Crisfield
c       dl0 = 1.d0
c       dlmax = 1.
c       dlmin = .1
```

```
c      exlmx = 100.d0
c     toldis = .99d0
c     tolfor = .99d0
C*************************************************************
c
C*************************************************************
C** The following are used for test problelm T11in.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
c       dl0 = 1.d0
c       dlmax = .1
c       dlmin = .01
c      exlmx = 1000.d0
c     toldis = .9d0
c     tolfor = .98d0
C*************************************************************
C*************************************************************
C** The following are used for test problelm T11in.dat
C** This is a 4 ele, 4 node problem, results agree with Crisfield
c       dl0 = 1.d0
c       dlmax = .1
c       dlmin = .01
c      exlmx = 100.d0
c     toldis = .99d0
c     tolfor = .99d0
C*************************************************************
c ... end user input
c

c                     allocate (dqtot(ndof),dqi(ndof),dqiml(ndof),qk(ndof))

c                     If(iopt .eq. 1)then
c                     deallocate (dqtot)
c                     deallocate (dqi)
c                     deallocate (dqiml)
c                     print *,'deallocate'
c                     endif

                      allocate (dqi(ndof),dqiml(ndof))

                      ncoef1_guess=20000000
      allocate ( itempo2(ncoef1_guess) )

c                     print *,'check BB'
c                     pause

c     common/RIK/ stifi(60,60),dqk1(5000),
c     $     dqk2(5000),fintk(5000),
c     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c      allocate ( ie(ieall+1),je(jeall),iboundc(ndof),iet(ndof+1),
c     $     jet(jeall),ip(ndof),iu(ndof+1),iut(ndof+1),iup(ndof),
c     $     isupn(ndof),lm(maxdofpe),iperm(ndof),ia(ndof+1),iarea(ndof))
c                      allocate (ja(ndof+1))
c     allocate ( b(ndof),be(maxdofpe),ae(maxdofpe**2),ad(ndof),
c     $     an(10000),dqtot(ndof),
c     $     x(numnodes),y(numnodes),z(numnodes),
c     $     propmat(nummat*11),propsect(nsect),tempo1(ndof),
c     $     tempo2(ndof),di(ndof),exfor(ndof))
c      allocate ( disp(ndof),qk(ndof) )

c                     Do 7070 i=1,jeall
c                     je(i)=0.d0
c7070  continue

      itnrmlon = 0

                      Do 7070 i=1,maxdofpe**2
                      ae(i)=0.d0
7070  continue

c                     Do 6060 i=1,ndof+1
c                     ia(i)=0.d0
c                     ja(i)=0.d0
c6060  continue

                      Do 5050 i=1,10000
                      an(i)=0.d0
5050  continue

                      Do 4040 i=1,1000
                      ja2(i)=0.d0
4040  continue

                      Do 3030 j=1,60
                      Do 3031 k=1,60
                      stifi(j,k)=0.d0
3031  continue
3030  continue

                      Do 2020 i=1,5000
                      dqk1(i)=0.d0
                      dqk2(i)=0.d0
                      fintk(i)=0.d0
                      resk(i)=0.d0
                      scali(i)=0.d0
                      work1(i)=0.d0
                      indx(i)=0.d0
2020                  continue

                      Do 1010 i=1,1500
                      a(i)=0.d0
1010                  continue

c ... initialize required variables
c
      dli = 0.d0
                      dlk = 0.d0
      dliml = 0.d0
      dqsq = 0.d0
      ds0 = 0.d0
      dsi = 0.d0
      dsiml = 0.d0
      dsmax = 0.d0
      dsmin = 0.d0
      exlk = 0.d0
      iconv = 0
      istep = 1
      it = 1
c
      do 20 i=1,neq
c        do 10 j=1,neq
c                     print *,'chk before stifi'
c                     pause
c          stifi(j,i) = 0.d0
c 10     continue
c
c                     print *,'check zz'
c                     pause
      dqi(i) = 0.d0
c                                   print *,'Check after dqi'
c                     pause
      dqiml(i) = 0.d0
      dqtot(i) = 0.d0

c                       disp(i) = 0.d0

                        fintk(i) = 0.d0
      qk(i) = 0.d0
      resk(i) = 0.d0
      scali(i) = 0.d0
      work1(i) = 0.d0
c                       exfor(i) = 0.d0
                        b(i) = 0.d0
                      disp(i)=0.d0
                      ad(i)=0.d0
c                     tempo1(i)=0.d0
c                     tempo2(i)=0.d0
 20   continue

c     print *,'exfor at start rikarc = ',(exfor(ic),ic=1,ndof)
c                     pause
c                     print *,'iarc=',iarc
c                     print *,'maxdofpe=',maxdofpe
c                     print *,'iboundc=',(iboundc(i),i=1,ndof)
c                     pause

c      stop
c
      iskip = 0 !When zero will go thru symfactd, 1 will skip symfactd

      itnrmlon = 0 !When zero not going thru "itnrml"
c               When one, inside "itnrml" telling sysqn & assembly
c
c... [ja] allocation in sysqn..if jallo=0 then allocate
c... if jallo=1 then skip allocation is sysqnF

                      If(iopt .eq. 0)then
c                     print *,'iopt is zero at start Rikarc=',iopt
c                     pause
                      jallo=0
                      endif

c                     print *,'exlk before 999=',exlk
c                     pause

 999  if (exlk .le. exlmx .and. istep .le. mxstp) then

c                       If(istep.eq.910)Then
c                           itedes=1
c       dl0 = .01d0
c       dlmax = .000001d0
c       dlmin = .000000001d0
c       exlmx = 1.d0
c       toldis = 1.d-8
c       tolfor = 1.d-8
c                       print *,'Change starting arc varibles'
c                       Endif

c     write(*,*) 'Beginning Load Step Number in RIKARC:',istep
c                       print *,'exlk start Rik=',exlk
c      pause
c      stop
      it = 1
      if (istep .eq. 1) then
c     print *,'** CHECK A **'
c                       pause
c                       print *,'je start istep=1  = ',(je(i),i=1,20)

c      print *,'getstiff=',getstiff
c      print *,'getforce=',getforce
c      print *,'getsolve=',getsolve
c      pause

c                       stop
c ... compute Kt0 for eqn. 4.107
c                       print *,'ndof=',ndof
c                       pause
c                       print *,'b before syseqn = ',(b(i),i=1,ndof)
c                       pause
c                       stop
c                       print *,'maxdofpe=',maxdofpe
c                       print *,'check before 1st syseqn'
c
c     stop
c                       getstiff=1

      call sysqn(nc,tiplvl,b,ndof,disp,iarc,qk,
     $           ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $           neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $           iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqiml,
     $          ux,uy,uz,di,it,area,youngm,iarea,
     $         ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $         jallo,
     $         strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
```

```
$        Fin2,VED,VEDold,xlie,SED)

c                                     write(6,*)'ia2 after syseqn=',(ia2(i),i=1,ndof+1)
c   $ja,iboundc,ndofped,bc,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c    print *,'chk after syseqn in rikarc'
c                   print *,'*******************'
c                   print *,'*******************'
c                   print *,'disp after 1st syseqn=',(disp(i),i=1,ndof)
c                   print *,'*******************'
c                   print *,'*******************'
c                   pause
c                   print *,'iskip=',iskip
c                   print *,'itempo2(1) after 1st syseqn=',itempo2(1)
c                   pause

c                    print *,'b after 1st syseqn=',(b(i),i=1,ndof)
c                   pause
c                   stop
c                   write(6,*),'disp after 1st syseqn=',(disp(i),i=1,ndof)
c                   write(6,*),'ae stiff after 1st syseqn=',(ae(i),i=1,maxdofpe**2)
c                   print *,'ad stiff after 1st syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                                  print *,'an stiff after 1st syseqn=',(an(ic),ic=1,50)
c                   print *,'ae stiff after 1st syseqn=',(ae(ic),ic=1,maxdofpe**2)
c                   pause
c                   stop

        do 704 i=1,neq
c            scali(i) = 1.d0/stifi(i,i)
c     c  c      print *,'a(neq*2+i) = ',a(neq*2+i)
cc              scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
            scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c                   print *,'scali =',scali(i)
 704       continue

c                   stop
c    print *,'a after syseqn = ',a
c         stop
        getstiff = 0
        getforce = 0
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c      print *,'a = ',a
c      stop

c      stop

     do 44 i=1,ndof
44   dqtot(i)=disp(i)
c    print *,'dqtot(i) = ',(dqtot(i),i=1,ndof)
c                   pause
c                   write(6,*)'dqtot( )=',(dqtot(i),i=1,ndof)
c      stop
c ... compute dqtot**2 for eqn. 4.109
        dprd1 = 0.d0
        do 50 i=1,neq
          dprd1 = dprd1 + dqtot(i)**2
c    print *,'dprd1 = ',dprd1
 50       continue
c     write(6,*)'dprd=',dprd
c    print *,'dprd1 = ',dprd1
c                   pause
c      stop
c ... compute ds0 using eqn. 4.109
      ds0 = dl0*dsqrt(dprd1+1.d0)
c ... compute dsmax using user specified max load inc
        dsmax = dlmax*dsqrt(dprd1+1.d0)
c ... compute dsmin using user specified min load inc
        dsmin = dlmin*dsqrt(dprd1+1.d0)
c ... compute and store ds(i-1) for eqn. 4.110
        dsim1 = ds0

c    print *,'dsmax = ',dsmax
c    print *,'dsmin = ',dsmin
c    print *,'dsim1 = ',dsim1
c                   pause
c         stop

c ... compute dq0 using eqn. 4.106, and store in dqi
c ... store dq0 in dqim1 for next load step
c ... update the total displacements qk
c c     print *,'dl0 & dqtot(i) =',dl0,dqtot
c    print *,'neq = ',neq

c                    write(6,*)'qk(i) before=',(qk(i),i=1,neq)

c                   print *,'dl0=',dl0
c                   print *,'dqtot(_)=',(dqtot(ic),ic=1,neq)
c                   pause

        do 60 i=1,neq
          dqi(i) = dl0*dqtot(i)
c    print *,'dqi(i) = ',dqi
          dqim1(i) = dqi(i)
c    print *,'qk(i)before=',qk
c          disp(i) = disp(i) + dqi(i)
c                        print *,'dqi(i)=',dqi(i)
c                        print *,'disp(i) = disp(i) + dqi(i)',disp(i)
c                        pause
          qk(i) = qk(i) + dqi(i)
c    print *,'qk(i)after=',qk(i)
c                   pause
 60       continue
c                    write(6,*)'dl0=',dl0
c                    write(6,*)'dqtot(i)=',(dqtot(i),i=1,neq)
c                    write(6,*)'dqi(i)=',(dqi(i),i=1,neq)
c                    write(6,*)'qk(i)=',(qk(i),i=1,neq)

c         stop
c    print *,'dqtot(i) = ',dqtot
c    print *,'qk(i) = ',qk
c                   pause
c                   stop

c?? not sure ??
c** Put qk in "a"
ccc       Do 61 iqk = 1,neq
ccc       a(iqk) = qk(iqk)
ccc 61     continue

c** Put qk in "disp"
      Do 61 iqk = 1,neq
      disp(iqk) = qk(iqk)
 61    continue
c                   print *,'@@@@@@@@@@@@@@@@@@@@@@@@'
c                   print *,'disp(i)=qk(k)',(disp(ic),ic=1,neq)
c                   print *,'@@@@@@@@@@@@@@@@@@@@@@@@'
c                   pause

c    print *,'"a after insert qk" =',a
c      stop
c ... update the total external load multiplier exlk
                   If(exlk .GT. .98)then
                   print *,'after exlk 1 chk'
                   print *,'resk(_)=',(resk(ic),ic=1,neq)
                   pause
                   endif

                    IF(exlk .GT. exlk+dl0)then
                   exlkdl0=exlk+dl0
                   icxlk=1
c                   print *,'After 1st exlk check'
c                   print *,'exlk=',exlk
c                   print *,'exlk+dl0=',exlkdl0
c                   print *,'ad(i) after exlk=',(ad(i),i=1,ndof)
c                   pause

c                    print *,'After 1st exlk check'
c                   print *,'resk(_)=',(resk(ic),ic=1,neq)
c                   pause
c                   deallocate (dqtot)
c                   deallocate (dqi)
c                   deallocate (dqim1)
                    GOTO 6000
c                   stop
c                    pause
                    ENDIF
        exlk = exlk + dl0
                        exlkdl0=exlk
c     print *,'exlk=',exlk
c                    print *,'After 1st exlk check'
c                   pause

c ... store dl0 in dlim1 for next load step
        dlim1 = dl0
c                   print *,'dlim1=',dlim1
c                   pause
c ... compute the internal forces, fintk, at qk
        getforce = 1 !!!Yes get internal force
c     print *,'getforce = ',getforce
c                   write(6,*)'getforce=',getforce
c                   write(6,*)'exlk=',exlk
c                   write(6,*)'dl0=',dl0
c                    write(6,*)'b before 2nd syseqn=',(b(i),i=1,ndof)
c         stop
c**************************************************************
c        print *,' "a" before 2nd syseqn to getforce = ',a
c         stop
c                     print *,'it before 2nd syseqn in RIK=',it
c                     print *,'before assembly instead of call to syseqn'
c                     print *,'ja2(i) before assembly=',(ja2(ic),ic=1,10)
c                     print *,'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&'
c                     pause
c                         call assembly(ip,ad,an,lm,ic,je,propmat,
c    $         propsect,elk,be,b,iboundc,
c    $         ae,ia,ja,x,y,z,ncoef1,maxdofpe,
c    $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c    $ tempo1,disp,iarc,it,area,youngm,esm)

c                   print *,'*******************'
c                   print *,'*******************'
c                   print *,'disp before 2nd syseqn=',(disp(i),i=1,ndof)
c                   print *,'*******************'
c                   print *,'*******************'

c                    print *,'b before 2nd syseqn=',(b(i),i=1,ndof)
c                    pause

c                   print *,'dqtot before 2 syseqn=',(dqtot(i),i=1,neq)
c                   pause
c    print *,'qk(i) before 2 syseqn =',(qk(i),i=1,neq)
c                   pause
c                   stop

        call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
    $            ieall,ie,jc,ict,jct,jeall,ia,ae,maxdofpe,
    $            neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
    $            iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
    $          ux,uy,uz,di,it,area,youngm,iarea,
    $         ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
    $         jallo,
    $         strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
    $         Fin2,VED,VEDold,xlie,SED)

c    $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c                    print *,'Chk after 2nd syseqn'
c                   pause
c                   print *,'*******************'
c                   print *,'*******************'
c                   print *,'disp after 2nd syseqn=',(disp(i),i=1,ndof)
c                   print *,'*******************'
c                   print *,'*******************'
c                   pause
c                   stop

c                   print *,'ad stiff after 2nd syseqn=',(ad(i),i=1,maxdofpe**2)
```

```
c                    pause
c                                       print *,'an stiff=',(an(i),i=1,30)
c                    print *,'ae stiff after 2nd syseqn=',(ae(i),i=1,maxdofpe**2)
c                    pause

c                    stop
c                     print *,'qk(-)after 2nd syseqn=',(qk(i),i=1,ndof)

c                     print *,'b after 2nd syseqn=',(b(i),i=1,ndof)
c                    pause
c
c                     write(6,*)'b after 2nd syseqn=',(b(i),i=1,ndof)
c                     write(6,*)'Chk after 2nd syseqn'

c                     stop
c     print *,'Check after 2nd syseqn'
c                    stop
c     print *,'getforce = ',getforce
c     print *,'getstiff =',getstiff
c      print *,'a after getforce=',a
c      print *,'exlk after syseqn = ',exlk
c                     print *,'exfor before resid=',(exfor(i),i=1,neq)
c                     stop
ccc         do 705 i=1,neq
c             scali(i) = 1.d0/stifi(i,i)
c     c  c      print *,'a(neq*2+i) = ',a(neq*2+i)
cc              scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
ccc               scali(i) = 1.d0/ad(i)  !!!!Get diagonal

ccc                  print *,'scali =',scali(i)
ccc 705       continue

c                     stop
c                    print *,'dqtot before resid=',(dqtot(i),i=1,neq)
c                    pause
c     print *,'qk(i) before resid =',(qk(i),i=1,neq)
c                    pause
c                    stop
       getforce=0 !set back to zero
c*******************************************************
        call resid(dqtot,dqi,dqim1,qk,b,exfor)
c      print *,'resk after 1st resid = ',(resk(ic),ic=1,neq)
c                     pause
c      print *,'exlk after resid = ',exlk
c          stop
c ... test for convergence

c          call test(toldis,tolfor,exlk,iconv)
c                     print *,'Check before test 1'
c                    stop
       call test(dqtot,dqim1,qk,exfor,b)
c      print *,'iconv after test = ',iconv
c      print *,'stop after test'
c      print *,'istep = ',istep
c      print *,'exlk = ',exlk
c                    pause
c          stop

       if (iconv .eq. 0) then
c                    print *,'resk(3) result1=',resk(3)
c                    pause
          call result(istep,it,dqtot,dqim1,qk,exfor,propsect,ad,
   $            strainel,strainE,strainD,weight,nel,Anew,Fin1,
   $            xlie,weightotal,Nevenodd,N)

c    $             strainel,strainE,strainD)
c                    If(istep .eq. 1)then
   print *,'after result and istep 1'
c                    pause
c                    Endif

          istep = istep + 1

          stop
          go to 999
       end if
c         istep = istep + 1
c     print *,'exlk right before else = ',exlk
c      print *,'a before else = ',a
c       stop
c))))))))))))))))))))))))))))))))))))))))))))))))))))
      else
c     print *,'** CHECK B **'
c                    print *,'istep in RIK=',istep
c                    pause
c     print *,'** CHECK B **'
c                    print *,'istep in RIK=',istep
c                    pause
c     print *,'** CHECK B **'
c                    print *,'istep in RIK=',istep
c                    pause
c                    print *,'ia(_)=',(ia(ic),ic=1,ndof)
c                    pause
c       stop
c))))))))))))))))))))))))))))))))))))))))))))))))))))
c ... compute dsi from eqn. 4.110
cc      print *,'after else '
        dsi = dsim1*dsqrt( dfloat(itedes)/dfloat(istep) )
c         dsi = dsim1*dsqrt( dfloat(itedes)/dfloat(it) )
c     print *,'Starting 2nd part of Rikarc'
c     print *,'dsi = ',dsi
c                    print *,'dsmax=',dsmax
c                    print *,'dsmin=',dsmin

c                    pause
c         stop
c     print *,'exlk after else =',exlk
c       stop
c ... test to see if dsi is between dsmax and dsmin
        if (dabs(dsi) .gt. dabs(dsmax)) then
          if (dsi .gt. 0.d0) then
            dsi = dsmax
c                    print *,'dsi check dsmax=',dsmax
          else
c         c     write(*,*) 'Program terminated!!'
c         c     write(*,*) 'Magnitude of tangent vector is negative'
            stop
          end if
        elseif (dabs(dsi) .lt. dabs(dsmin)) then
          if (dsi .gt. 0.d0) then
            dsi = dsmin
c                    print *,'dsi check dsmin=',dsi
          else
c         c     write(*,*) 'Program terminated!!'
c         c     write(*,*) 'Magnitude of tangent vector is negative'
            stop
          end if
        end if
c                    pause
c ... store dsim1 for next load step
        dsim1 = dsi
c ... compute Kti for eqn. 4.113

c     print *,'stop before syseqn'
c          stop


c          call stiff
c**Only get stiffness
      Getstiff=1
                    Getforce=0
                    Getsolve=0
c                    print *,'getstiff=',getstiff
c                    print *,'getforce=',getforce
c                    print *,'ja(_)=',(ja(ic),ic=1,3)
c                    print *,'check before 3rd syseqn'
c                    stop
c                    pause
      call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
   $          icall,ic,jc,iet,jet,jeall,ia,ae,maxdofpe,
   $          neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
   $          iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
   $          ux,uy,uz,di,it,area,youngm,iarea,
   $      ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
   $      jallo,
   $      strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
   $      Fin2,VED,VEDold,xlie,SED)

c     Sja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c**I call modify to get the stiffness w/delected rows/columns

c                    print *,'ia(_)=',(ia(ic),ic=1,ndof)
c                    pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                    stop
c                    pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                    pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                    pause
c     print *,'********* after 1st syseqn after Else in Rikarc ******'
c                    pause
c      stop

cc       call modify
c     print *,'Check after third modify'
c     print *,'a after 3rd modify =',a
c     print *,'exlk after 3rd modify =',exlk
c       stop
c  c     pause
c c     print *,'ibstor = ',ibstor
c c     print *,'bvstor = ',bvstor
      Getstiff = 0 ! set getting just stiff back to zero
c                    print *,'getstiff=',getstiff
c                    print *,'getforce=',getforce
c                    print *,'getsolve=',getsolve
c                    pause
c       stop
c ... store scale factor for convergence testing, eqn. 4.133
c ... store scale factor for convergence testing, eqn. 4.133
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c      If(ityp .eq.  6)Then
c                    print *,'ad(_)=',(ad(i),i=1,neq)
c                    stop
        do 70 i=1,neq
c             scali(i) = 1.d0/stifi(i,i)
c     c  c     print *,'a(neq*2+i) = ',a(neq*2+i)
cc              scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
             scali(i) = 1.d0/ad(i)  !!!!Get diagonal
c              scali(i) = dabs(1.d0/ad(i))  !!!!Get diagonal
c                    print *,'scali =',scali(i)
 70       continue
c                    pause
c                    stop
c      Endif
c      If(ityp .eq.  12)Then
c         do 71 i=1,neq
c             scali(i) = 1.d0/stifi(i,i)
c     c  c     print *,'a(neq*2+i) = ',a(neq*2+i)
c              scali(i) = 1.d0/a(neq*3+i)  !!!!Get diagonal
c 71        continue
c      Endif
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

c     print *,'scali(i) check = ',(scali(kk),i=1,neq)
c          stop
c          do 70 i=1,neq
c              scali(i) = 1.d0/stifi(i,i)
c 70        continue
c ... solve eqn. 4.113 for dqtot
        do 80 i=1,neq
          dqtot(i) = exfor(i)
 80       continue
```

```
c      print *,'exfor(_)=',(exfor(ic),ic=1,ndof)
c                    pause
c**Only solve
       Getstifl=0
                      Getforce=0
                      Getsolve=1
c                    print *,'check before syseqn 4'
c                    stop
      call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $            ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $            neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $            iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $            ux,uy,uz,di,it,area,youngm,iarea,
     $        ianal,lumpmass,iskip,itempo2,itmrmlon,exfor,istep,
     $        jallo,
     $        strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $        Fin2,VED,VEDold,xlie,SED)

                      Getsolve=0


                      Do 82 i=1,neq
                      dqtot(i)=disp(i) !get new disp(_) & set equal to dqtot(_)
82                    Continue

c      print *,'dqtot(_)= ',(dqtot(ic),ic=1,neq)
c                     pause
c      stop
c put dqtot(i) into "a(i)" in place of force
       jay=0
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c      If(ityp .eq. 6)Then
c        do 81 i=neq+1,neq*2
         do 81 i=1,neq
            jay=jay+1
c          a(i)=dqtot(jay)
          b(i)=dqtot(jay)
c                     print *,'b(i)=',b(i)
 81       continue
c                     pause
c      Endif
c       If(ityp .eq. 12)Then
c        do 82 i=neq+1,neq*3
c           jay=jay+1
c          a(i)=dqtot(jay)
c 82      continue
c      Endif
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
cc     print *,'a w/dqtot=',a
c        stop
c        call solve(dqtot)
cc  c    print *,'stifi before ludcmp = ',stifi
c         call ludcmp(stifi,n,np,indx,x)
cc  c    print *,'stifi after ludcmp = ',stifi
c         stop
c         call lubksb(stifi,n,np,indx,dqtot)
cc  c    print *,'stifi after ludcmp = ',stifi

c      write(*,*) 'Before dcmpbd inside rikarc'
c      stop
      ipick = 3
cc     call dcmpbd
c     stop
c      write(*,*) 'After dcmpbd inside rikarc'
c      stop
cc      call slvbd
c      write(*,*) 'After slvbd & ipick=3 inside rikarc'
cc     print *,'a after slvbd=',a
c      stop

c ... compute dqtot**2 for eqn. 4.109
        dprd1 = 0.d0
        do 90 i=1,neq
          dprd1 = dprd1 + dqtot(i)**2
c           dprd1 = dprd1 + a(i)**2
 90      continue
c ... dli using eqn. 4.112
        dli = dsi/dsqrt(1.d0 + dprd1)
c                     If(dprd1.lt.0.)Then
c       print *,'dli inside RIK=',dli
c                     stop
c                     Endif
c       pause
c       stop
c ... test for sign of dli using eqn. 4.117
        dprd2 = 0.d0
        do 100 i=1,neq
          dprd2 = dprd2 + dqtot(i)*dqim1(i)
c           dprd2 = dprd2 + a(i)*dqim1(i)
c           dprd2 = dprd2 + b(i)*dqim1(i)
 100      continue
c                     print *,'istep=',istep
c                     print *,'dqtot(_)=',(dqtot(ic),ic=1,neq)
c                     print *,'dqim(_)=',(dqim1(ic),ic=1,neq)
c                     pause

                      If(dprd2.lt.0.)Then
c       print *,'dprd2 negative=',dprd2
c                      pause
c                     stop
                      Endif

                      If(dlim1.lt.0.)Then
c                     print *,'istep=',istep
c     print *,'dlim1 negative=',dlim1
c                     pause
c                     stop
                      Endif

c     print *,'dprd1=',dprd1
c     print *,'dprd2=',dprd2
c                     pause
c      stop

        testv = dli*(dprd2 + dlim1)
c                     print *,'testv=',testv
c                     print *,'istep=',istep
c                     pause



cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        if (testv .lt. 0.D0) then
c        if (testv .le. 0.D0) then
         dli = -1.d0*dli
c          dli = 1.d0*dli
c          print *,'test for negative sign'
c                pause
c          stop
cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        else
          dli = 1.d0*dli
c          dli = -1.d0*dli
c          print *,'test for positive sign'
c          stop
        end if

c                                    If(istep .eq. 914)then
c                print *,'it=',it
c                pause
c                                    If(it .gt. 4)then
c                                    If(ad(6) .lt. 0.)then
c                                    dli = -1.d0*dli
c                   exlk=-exlk
c                                    Endif

c    pause

c ... compute dqi using eqn. 4.114
c ... store dqi in dqim1 for next load step
c ... update the total displacements qk
        do 112 i=1,neq
          dqi(i) = dli*dqtot(i)
c          dqi(i) = dli*a(i)
c          dqi(i) = dli*b(i)
          dqim1(i) = dqi(i)
          qk(i) = qk(i) + dqi(i)

c** Need to put qk into "a" at displacement locations
c           a(i) = qk(i)
c                    b(i) = qk(i)
                     dqtot(i)=qk(i)
 112      continue
c     print *,'qk(i) = ',qk
cc     print *,'a w/qk update added =',a

c      stop
c ... update the total external load multiplier exlk
c     print *,'exlk before update =',exlk
c                    pause
c      stop
c     If(istep .eq. 561) then
c                    dli=abs(dli)
c                    endif
                     If(exlk .GT. .98)then
c                    print *,'after exlk 2 chk'
c                    print *,'resk(_)=',(resk(ic),ic=1,neq)
c                    pause
                     endif

                      IF(exlk .GT. exlk+dli)then
                     exlkdli=exlk+dli

                     iexlk=2

c                     print *,'After 2nd exlk chock'
c                    print *,'exlk=',exlk
c                    print *,'dli=',dli
c                    print *,'exlk+dli=',exlkdli
c                    print *,'ad(i) after exlk=',(ad(i),i=1,ndof)
c                    pause

c                    print *,'dli=',dli
c                    print *,'exlk=',exlk
c                     print *,'After 2nd exlk chock'
c                    print *,'resk(_)=',(resk(ic),ic=1,neq)
c                    pause
c                    pause

c                    deallocate (dqtot)
c                    deallocate (dqi)
c                    deallocate (dqim1)
c                    deallocate (exfor)
c                    deallocate (qk)
c                    deallocate (ad)

                      GOTO 6000
c                     pause
c                      STOP
                      ENDIF
        exlk = exlk + dli

c                     If(istep .eq. 561) then
c                      exlk=-exlk
c                     endif
c     print *,'dli = ',dli
c     print *,'exlk after update =',exlk
c       stop
c c    print *,'exlk BBBBBBBB = ',exlk
c       stop
c ... store dli in dlim1 for next load step
        dlim1 = dli
c ... compute the internal forces, fintk, at qk
c         call forces
        getforce = 1 !!!Yes get internal force
c c     print *,'getforce = ',getforce
```

```
c          stop
c*************************************************************
c c     print *,' "a" beforc 2nd sysoqn to getforce = ',a
c                print *,'ia(_)=',(ia(ic),ic=1,ndof)
c                pause
c     print *,'********* after 2nd sysoqn after Else in Rikarc ******'
c                pause
c     print *,'********* after 2nd syseqn after Else in Rikarc ******'
c                pause
c     print *,'********* after 2nd syseqn after Else in Rikarc ******'
c                pause
c     print *,'********* after 2nd syseqn after Else in Rikarc ******'
c                pause
c       stop
c          stop
        call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $              ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $              neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $              iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $              ux,uy,uz,di,it,area,youngm,iarea,
     $      ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $      jallo,
     $      strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $      Fin2,VED,VEDold,xlie,SED)

c    $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)

c c     print *,'Check after 2nd syseqn'

c^^^^^ SETTING ZERO DIAGONAL VALUES TO ONE ^^^^^^^^^^^^^^^^
c       Do 1013 i=neq*2+1,neq*3
c                 Do 1013 i=1,neq
c       If(a(i) .GT. .01) GoTo 1013
cc       If(ad(i) .GT. .000001) GoTo 1013
c       If(ad(i) .GT. 1.e-8) GoTo 1013
c         a(i)=1.0
c                 ad(i)=1.e-8
c                 ad(i)=0.0
c1013    Continue
         do 701 i=1,neq
c              scali(i) = 1.d0/stifi(i,i)
c     c c      print *,'a(neq*2+i) = ',a(neq*2+i)
cc             scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
               scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c                print *,'scali =',scali(i)
 701      continue

c c     print *,'getforce = ',getforce
c     print *,'a=',a
c        stop
cc        call modify
c     print *,'Check after forth modify'
c                pause
c                stop
c c     print *,' "a" after 4th modify = ',a
c     c     pause
         getforce=0 !set back to zero
c c     print *,' "a" after getforce = ',a
c       stop
c     print *,'Before 2nd resid'
c                 Stop
c ... compute the residual forces, resk, at qk and exlk
         call resid(dqtot,dqi,dqim1,qk,b,exfor)
c     print *,'resk(_) after 2nd resid in itnrml= ',(resk(ic),ic=1,neq)
c                pause
c     print *,'exlk after 2nd resid = ',exlk
c                pause
c c     print *,'After resid'
c          stop
c ... test for convergence
c          call test(toldis,tolfor,exlk,iconv)
c                print *,'Before test 2'
          call test(dqtot,dqim1,qk,exfor,b)
c     print *,'After test'
c                 print *,'iconv after test = ',iconv
c     print *,'istep = ',istep
c                pause
c          stop
          if (iconv .eq. 0) then
c     print *,'exlk before result=',exlk

c          stop
c                print *,'resk(3) result2=',resk(3)
c                pause

           call result(istep,it,dqtot,dqim1,qk,exfor,propsect,ad,
     $               strainel,strainE,strainD,weight,nel,Anew,Fin1,
     $               xlie,weightotal,Nevenodd,N)

c                 If(istep .eq. 1)then
c                 print *,'after 2nd "call result" in rikarc'
c                pause
c                 Endif

           istep = istep + 1
c c  c     print *,'istep = ',istep
c     print *,'Ready to go back to 999'
c     pause
c           stop
           go to 999
          end if
c)))))))))))))))))))))))))))))))))))))))))))))))))))))
        end if
c)))))))))))))))))))))))))))))))))))))))))))))))))))))
cc      print *,'"a" before itntml = ',a
c      print *,'ready to go into itnrml'
c     pause
c      print *,'ready to go into itnrml'
c     pause
c      print *,'ready to go into itnrml'
c     pause
c      print *,'ready to go into itnrml'
c     pause

c          stop
c
c       call itnrml(itemax,toldis,tolfor,dli,dlim1,
c     $                 exlk,iconv,iflag,it,ne,tplvl)
c     print *,'exlk right before itnrml = ',exlk
c       stop
c     print *,'exlk right before itnrml = ',exlk
c     print *,'dli before itnrml = ',dli
c     print *,'dqi before itnrml = ',dqi
c     print *,'a before itnrml = ',a
c      print *,'exfor before itnrml=',exfor
c       stop
c                 pause
c                 print *,'********************'
c                 print *,'********************'
c                 print *,'disp before itnrml=',(disp(i),i=1,ndof)
c                 print *,'********************'
c                 print *,'********************'
c     print *,'exfor before itnrml= ',(exfor(ic),ic=1,ndof)
c                 pause
c                 stop

c                 print *,'ad stiff before itnrml=',(ad(ic),ic=1,maxdofpe**2)
c                 pause
c                 print *,'an stiff before itnrml=',(an(ic),ic=1,50)
c                 print *,'b(_) before itnrml=',(b(ic),ic=1,ndof)
c                 pause
c                 print *,'ae(_)before itnrml=',(ae(ic),ic=1,18)
c                 pause

c                 pause
c     print *,'ia(-) before itnrml = ',(ia(i),i=1,ndof+1)
c                 PAUSE
c                 stop
       call itnrml(itemax,dli,dlim1,b,exfor,ad,neltype,qk,
     $          iflag,it,ne,iplvl,ndof,ia,nel,ncoefl,jcall,
     $            ieall,ie,je,iet,jet,maxdofpe,dqtot,dqi,dqim1,
     $            ja,an,propmat,propsect,x,y,z,ndofpn,numnodes,
     $            ip,iu,disp,ianal,ndofpe,iskip,itempo2,iarc,lm,
     $            itnrmlon,iboundc,ae,di,istep,iarea,
     $          strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $          Fin2,VED,VEDold,xlie,exlkdl0,exlkdli,exlkdlk,iexlk)

                 itnrmlon=0 !set back to zero because we are done going thru "itnrml"

c     print *,'exfor after itnrml= ',(exfor(ic),ic=1,ndof)
c                 pause
c     print *,'Check after itnrml'
c                 pause
c     print *,'exlk after itnrml = ',exlk
c       stop
c     print *,'iflag = ',iflag
c     print *,'exlk right after itnrml = ',exlk
c     print *,'exfor(3) right after itnrml = ',exfor(3)
c     print *,'exfor(4) right after itnrml = ',exfor(4)
cc      print *,'a(10) right after itnrml = ',a(10)
c       stop
c     print *,'a after itnrml=',a
cc      print *,'ipick after itnrml=',ipick
c       stop
c
c     print *,'iconv after call to itnrml = ',iconv
                If(istep .eq. 1)then
c     print *,'istep = ',istep
c     print *,'exlk = ',exlk
c                 pause
                 Endif
c                 pause
c                 stop
       if (iflag .eq. 0) then
c      print *,'Check before result in itnrml'
c                  print *,'dqtot before result in itnrml=',(dqtot(i),i=1,6)
c                 print *,'istep before result in itnrml=',istep
c                 print *,'it before result in itnrml=',it
c         If(istep,it,dqtot,dqi,dqim1,qk,exfor)

c                 print *,'resk(3) result3=',resk(3)
c                 pause

       call result(istep,it,dqtot,dqim1,qk,exfor,propsect,ad,
     $               strainel,strainE,strainD,weight,nel,Anew,Fin1,
     $               xlie,weightotal,Nevenodd,N)

c                 If(istep .eq. 1)then
c                  print *,'after 3rd "call result" in rikarc'
c                  print *,'ad(3) =',ad(3)
c                 pause
c                 endif

c                     print *,'qk(_)=',(qk(ic),ic=1,ndof)
c                     print *,'disp(_)=',(disp(ic),ic=1,ndof)
c                     pause

c                 if(istep .eq. 255)then !write out stiffness diagonal
c                 if(istep .eq. 571)then !write out stiffness diagonal
c                  print *,'ad( ) stiff diag.',(ad(ic),ic=1,ndof)
c                 pause
c                 endif
c      print *,'STOP3 after result'
c                    pause
c        stop
       elseif (iflag .eq. 2) then
         write(6,*) 'MAX. ITERATIONS EXCEEDED IN ITNRML!!!!'
         write(6,*) 'PROGRAM TERMINATED !!!!!'
c     c     write(*,*) 'MAX. ITERATIONS EXCEEDED IN ITNRML!!!!'
c     c     write(*,*) 'PROGRAM TERMINATED !!!!!'
         go to 5000
       else
         write(6,*) 'UNKNOWN ERROR IN ITNRML!!!!'
```

```
        write(6,*) 'PROGRAM TERMINATING!!!!'
        go to 5000
      end if
c
cccc            if (istep .GE. 7) stop

      istep = istep + 1
c
c     print *,'Ready to go to 999 at end Rikarc'
c                 pause
c      stop
      go to 999
c
      end if
c
c     print *,'Before 5000 stop'
c                 pause
                  goto 6000
c                 dealloacate dqtot,dqi,dqim1,exfor,qk,ad

c     print *,'exlk before stop=',exlk
c     print *,'exlmx before stop=',exlmx
c     print *,'istep before stop=',istep
c     print *,'mxstp before stop=',mxstp
 5000  stop
 6000              return
      end
C*********************************************************************
*

c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      subroutine syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $          ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
c     $          ieall,ie,je,iet,jet,jeall,ia1,ia2,ae,maxdofpe,
     $          neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $          iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
c     $          iboundc,ad,elk,iperm,iu,ip,iut,dqtot,dqi,dqim1,
c     $          ux,uy,uz,di,it,area,youngm,esm,elk,
     $          ux,uy,uz,di,it,area,youngm,iarea,
     $          ianal,lumpmass,iskip,itempo2,itarmlon,exfor,istep,
     $          jallo,
     $          strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $          Fin2,VED,VEDold,xlie,SED)


c      call numfa1d(ndof,ia,ja,ad,an,iu,
c     $           ju,di,un,ip,iup,
c     $           isupn,iopf)

c     $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c**********
c,
c  generation of the system of equations
c
c**********
      implicit real*8(a-h,o-z)

can                   SAVE ja,an,itempo1
                      SAVE ja,itempo1
c                     SAVE itempo1
cccc      common/elmatx/esm(6,6),ef(6),x(6),y(6),z(6),kl,ndofpe
      common/element/ dx(5),dy(5),g(5),q(5),grdc(5,2)
      common/matl/dxe,dye,ge,qe
      common/hcv/idbc(500,2),dbc(500,2),ndbc
      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
c      common/nodes/nel(500,4),nmtl(500),xc(500),yc(500),zc(500)
      common/nodes/nmtl(500),xc(500),yc(500),zc(500)
c     dimension dx(5),dy(5),g(5),q(5),grdc(5,2)

c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c     $           dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c     $           resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c     $           dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c     $           resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(60,60),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c          common/deltaqi/dqi(500)

c     common/disp/ux(2),uy(2),uz(2),disp(500)

c     common/disp/ux(2),uy(2),disp(10)
c     common/disp/ux(2),uy(2)
      common ns(6),phi(6),gdx(6),gdn(6),gdy(6)
c     common/disp/qk(500)
      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
c     common/eletype/ityp  !ityp = 6 for 2-D problems
      common/spring/springK
      common/testing/toldis,tolfor,exlk,iconv

                      common/jam/ja2(2000)

c                     integer, allocatable:: ja(:),itempo1(:),itempo2(:)
ccccccccccccccccccccccccccc
                      integer, allocatable:: ja(:),itempo1(:),ju(:),jut(:)
cc                    integer, allocatable:: itempo1(:),ju(:),jut(:)
ccccccccccccccccccccccccccc
c                     integer, allocatable:: itempo1(:)

c                     integer, allocatable:: itempo1(:)
ccc     doublc precision, allocatable:: an(:)

c     double precision, allocatable:: tempo1(:),an(:),ju(:),jut(:),un(:)
      double precision, allocatable:: tempo1(:),un(:)
c     double precision, allocatable:: ju(:),jut(:),un(:)
                      dimension b(*),disp(*),ux(*),uy(*),uz(*),x(*),y(*),z(*)
c                     dimension ie(*),je(*),iet(*),jet(*),ia(*)
can                   dimension ie(*),je(*),iet(*),jet(*),ae(*),ad(*)
                      dimension ie(*),je(*),iet(*),jet(*),ae(*),ad(*),an(*)
                      dimension iu(*),ip(*),iut(*),iperm(*)
c                     dimension dqtot(*),dqi(*),dqim1(*),di(*),ia(*),qk(*)
                      dimension dqtot(*),dqim1(*),di(*),ia(*),qk(*)
c                     dimension dqtot(*),dqi(*),dqim1(*),qk(*),di(*),ia1(*),ia2(*)
c                     dimension iboundc(*),elk(maxdofpe,1),ja(*),an(*),lm(1)
                      dimension iboundc(*),lm(1)
c                     dimension iboundc(*),elk(maxdofpe,maxdofpe),lm(1)
                      dimension itempo2(*),exfor(*)

                      dimension dqi(5000),dqk(5000)

                      dimension strainel(*),strainE(*),strainD(*),weight(*),xlie(*)

                      dimension Anew(*),Apart(*),Fin1(*),Fin2(*),VED(*),VEDold(*),SED(*)

c                     dimension ja(*)

c     $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c                     dimension be(*),ad(*),dm(*),am(*)

c                     dimension ja(19)

c                     dimension tempo1(10)
c                     print *,'S T A R T  O F  S Y S E Q N'
c                     pause
c                     print *,'iskip at start syseqn=',iskip
c                     print *,'getstiff at start syseqn=',getstiff
c                     print *,'getforce at start syscan=',getforce
c                     print *,'getsolve at start syseqn=',getsolve
c                     pause
c                     print *,'qk(_) at start of syseqn=',(qk(ic),ic=1,neq)
c                     pause

c                     print *,'b(_) inside start syseqn=',(b(ic),ic=1,neq)
c                     pause
c                     print *,'disp(_) inside start syseqn=',(disp(ic),ic=1,neq)
c                     pause
c                     print *,'ad(_) inside start syseqn=',(ad(ic),ic=1,neq)
c                     pause
c                     print *,'iskip start syseqn=',iskip
c                     pause

c...Solver needed, no update to stiff or forces required
c                     print *,'getsolve start syseqn=',getsolve
c                     pause
                      If(getsolve .eq. 1)Then
                      Goto 1111
                      Endif

                      If(getforce .eq. 1)then
c                     print *,'getforce =1 start syseqn, goto 32'
c                     pause
      Do 28 i=1,ndof
 28    b(i)=0.
                      goto 32
                      endif
c                     print *,'check before allocate (tempo1)'
                      allocate (tempo1(ndof))

c                     print *,'itarmlon start syseqn=',itarmlon
c                     pause
c*******************************
c*                    If(itarmlon .eq. 1)then
c*                    do 27 i=1,neq
c*                    dqk1(i) = exfor(i)
c*                    dqk2(i) = resk(i)
c*                    print *,'dqk1(_)=',dqk1(i)
c*                    print *,'dqk2(_)=',dqk2(i)
c*27    continue
c*                    j=0 !????????
c*                    do 29 i=1,neq
c*                    b(i)=dqk1(i)
c*                    print *,'b(_) in syseqn & itarmlon=',b(i)
c*29                  continue
c*                    Endif
c*******************************
c                     pause

c                     print *,'ianal at start of syseqn =',ianal
c                     print *,'lumpmass at start of syseqn =',lumpmass
c                     stop

c                     print *,'x start syseqn =',(x(i),i=1,ndof)
c                     print *,'y start syseqn =',(y(i),i=1,ndof)
c                     print *,'z start syseqn =',(z(i),i=1,ndof)
c                     print *,'ndof=',ndof
c                     stop
c                     print *,'chk start of syseqn'
c                     pause
c                     print *,'iarc=',iarc
c                     print *,'iboundc=',(iboundc(i),i=1,ndof)
c                     print *,'it = ',it
c                     stop

c                     print *,'maxdofpe =',maxdofpe
c                     stop
c                     print *,'b start of syseqn = ',(b(i),i=1,ndof)
c                     pause
c                     print *,'je start of syseqn = ',(je(i),i=1,20)
c                     stop

c                     if(it .eq. 2)then
c                     print *,'it =2'
c                     stop
c                     endif

c                     print *,'getforce start syseqn=',getforce
c                     print *,'getstiff start syseqn=',getstiff
c                     stop
c                     pause

c                     if(it .eq. 2)then
```

```
c                   print *,'it =2'
c                   stop
c                   endif

c       If(getforce .eq. 1)then
c     print *,'getforce=1 at start of syseqn'
c                   stop
c** Get ready for internal forces, set all forces to zero
c*****************************************************
c       If(ityp .eq. 6)Then
c       Do 28 i=1,ndof
c 28    b(i)=0.
c*****************************************************
c        stop
c       goto 30
c       endif

c** Get ready for new stiffness terms,
c** set only stiff & force terms to zero
c*****************************************************
c       DO 29 i=1,ndof
c  29   b(i)=0.  !set loads to zero
c                   print *,'b = ',(b(i),i=1,ndof)
c                   stop
c                   ndof2=ndof**2
c                   print *,'ndof2=',ndof2
c                     Do 31 i=1,ndof2-10
c  31   an(i)=0.  !set stiff to zero
c                   print *,'chk 2 inside syseqn'
c                   stop

30                  continue !just a blank, no Do associated
c......input boundary dof
c      call supportdof(iboundc,nboundc,b,ndof,ia)
c                   print *,'b inside syseqn = ',(b(i),i=1,ndof)
c                   pause
c                   stop
c......
c                   print *,'getforce=',getforce
cvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
                    IF(getforce .eq. 1)then  !skip transa2d, symbass, copy_int
c "Go to assembly2"
c                   print *,'ja(1)=',ja(1)
c                   print *,'getforce goto 32'
c                   pause
c                   stop
c                   print *,'icall=',icall
c                   print *,'ndof=',ndof
c                   print *,'jet = ',(jet(i),i=1,ieall*4)
c                   print *,'iet = ',(iet(i),i=1,ndof+1)
c    print *,'ie = ',(ie(i),i=1,icall+1)
c                   print *,'je = ',(je(i),i=1,jeall)
c                   stop
                     goto 32
                    Endif
cvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
c                   if(it .eq. 2)then
c                   print *,'it =2'
c                   stop
c                   endif
c                   print *,'chk before transa2d'
c                   print *,'getstiff=',getstiff
c                   print *,'getforce=',getforce
c                   pause
c                   print *,'je inside syseqn before transa2d= ',(je(i),i=1,20)
c                   pause
c                   stop
c                   print *,'chk before transa2d'
c                   pause
      call transa2d(icall,ndof,ie,je,iet,jet)

cc32    continue !dummy line from "if(getforce .eq. 1) above
c      write(6,*) '*** passed transa2 ***'
c                   print *,'after transa2d'
c                   pause
c                   stop
c                   print *,'ndof=',ndof
c                   print *,'jeall=',jeall
c                   print *,'ieall = ',ieall
c     print *,'ie = ',(ie(i),i=1,ieall+1)
c                   print *,'je = ',(je(i),i=1,jeall)
c                   print *,'je = ',(je(i),i=1,ndof*4)
c                   print *,'iet = ',(iet(i),i=1,ndof+1)
c                   print *,'jet = ',(jet(i),i=1,jeall)
c                   print *,'jet = ',(jet(i),i=1,ieall*4)
c                   print *,'ia1 = ',(ia1(i),i=1,ndof+1)
c                   stop
c                   if(it .eq. 2)then
c                   print *,'it =2'
c                   stop
c                   endif

c                   If(getforce .eq. 1)then
c                   print *,'getforce =1 before symbass'
c                   print *,'ndof=',ndof
c                   stop
c                   endif
c......sparse "symbolic" assembly
      ncoef1_guess=20000000
c                    deallocate (itempo1)
c
      allocate ( itempo1(ncoef1_guess) )
c                   print *,'iskip before allocate itempo2=',iskip
c                   pause
c                    If(iskip .eq. 1)Then
c                      itempo1(1)=itempo2(1)
c       print *,'itempo1(1)=itempo2(1)=',itempo1(1)
c                    Else
c                    Endif
c
c                    print *,'ndof=',ndof
c                    stop
c                   print *,'chk before symbass'
c                   print *,'getstiff=',getstiff
c                   print *,'getforce=',getforce
c                   print *,'ieall before symbass=',icall
c                   print *,'jeall before symbass=',jcall
c                   print *,'ie before symbass=',(ie(ic),ic=1,icall+1)
c                   print *,'je inside syseqn before symbass=',(je(i),i=1,30)
c                   print *,'iet before symbass=',(iet(ic),ic=1,ieall+1)
c                   print *,'jet before symbass=',(jet(ic),ic=1,ieall+1)
c                   print *,'ia before symbass=',(ia(i),i=1,icall+1)
c                   print *,'ja before symbass=',(ja(i),i=1,ieall+1)
c                   stop
c                   pause
c                   print *,'iskip before if = ',iskip
c                   pause
                    If(iskip .eq. 1)Then !only do symbass one time, skip after 1st time
                    Go To 799
                    Else
      call symbass(ie,je,iet,jet,ndof,ia,
     $          itempo1,ncoef1,istep)
c                    iskip = 1!This causes symbass to be skipped, only need to go once
                    itempo2(1)=itempo1(1) !Need this because itempo1 gets re-allocated
c      call symbass(ie,je,iet,jet,ndof,ia1,ia2,
c     $          itempo1,ncoef1)
      Endif
799   CONTINUE !dummy line for abv "If" statement
c                   print *,'iskip after symbass = ',iskip
c                   print *,'*** After symbass ***'
c                   print *,'je after syseqn = ',(je(i),i=1,20)
c                   print *,'itempo1=',itempo1(1)
c                   print *,'itempo2=',itempo2(1)
c                   pause

c                   stop
c                   pause

c                   If(getforce .eq. 1)then
c                   print *,'getforce =1 before symbass'
c                   print *,'ncoef1 =',ncoef1
c                   stop
c                   endif

c                   print *,'ncoef1 =',ncoef1
c                   pause
c                   stop
c                   print *,'ieall = ',ieall
c     print *,'ie = ',(ie(i),i=1,ieall+1)
c                   print *,'je = ',(je(i),i=1,jeall)
c                   print *,'iet = ',(iet(i),i=1,ndof+1)
c                   stop
c                   print *,'jeall=',jeall
c                   print *,'ndof=',ndof
c                   print *,'ncoef1=',ncoef1
c                   print *,'jet after symbass= ',(jet(i),i=1,jcall)
c                   print *,'ia1 after symbass= ',(ia1(i),i=1,ndof+1)
c                   stop
c                   print *,'ia2 after symbass= ',(ia2(i),i=1,ndof+1)
c                   print *,'itempo1 after symbass= ',(itempo1(i),i=1,ncoef1)
c                   pause
c                   stop

c                   If(getforce .eq. 1)then
c                   stop
c                   endif

c                   if(it .eq. 2)then
c                   print *,'it =2'
c                   print *,'STOP'
c                   stop
c                   endif

c                   stop
c                   print *,'ncoef1=',ncoef1
c                   stop
c                   do 111 i=1,ncoef1_guess
c111                print *,'itempo1 = ',itempo1(ncoef1_guess)
c                   stop
c                   print *,'ncoef1 = ',ncoef1
c     stop
c
c      allocate ( ja2(ncoef1) )
c                   stop
c...not sure, "ja" is also needed in assembly2
c                   print *,' ??????????????????????????????'
c...itnrmlon means call was done in subroutine itnrml
c                   print *,'itnrmlon=',itnrmlon
c                   pause
                    If(itnrmlon .eq. 1)Then !skip allocation, already done
                    goto 790
                    Endif

c                   print *,'getstiff=',getstiff
c                   pause

                    If(getstiff .eq. 0)then
c... when jallo=0, allocate [ja]
c... don't allocate when jallo=1
c... added for muliple loads for dynamics

c                   print *,'jallo=',jallo
c                    If(jallo .eq. 0) then
cc                     If(iloadyn .eq. 1 .or. idyn .ne. 99)then
c                                 print *,'chk before allocate (ja)'
c                      print *,'jallo before allocate=',jallo
c                      pause
cc         allocate ( ja(ncoef1) )
c                                 print *,'chk after allocate (ja)'
c                      jallo=1
cc                     endif
c                    endif

c                    print *,'ncoef1=',ncoef1
```

```
c                    pause

                     If(iopt .eq. 1)then !skip allocation
c                    print *,'iopt=',iopt
c                    pause
                     goto 790
                     Endif
      allocate ( ja(ncoef1) )
c                    print *,'allocate -ja- when getstiff=0 inside syseqn'
c                    pause
c     allocate ( ja2(ncoef1) )
can       allocate ( an(ncoef1) )
c                    print *,'allocate -an- when getstiff=0 inside syseqn'
c                    pause
                     Endif
790                  Continue !dummy line

c                     allocate (tempo1(ndof))
c                    print *,'***chk before copy_int***'
c                    pause
c                    print *,'getstiff=',getstiff
c                    print *,'getforce=',getforce
c                    print *,'getsolve=',getsolve
c                    print *,'ncoef1=',ncoef1
c                    print *,'itempo1=',itempo1
c                    print *,'ja=',(ja(ic),ic=1,ncoef1)
c                    pause
c                    print *,'iskip before 1st copy_int=',iskip
c                    pause

c                    print *,'iarea(_) before 1st copy_int=',(iarea(i),i=1,nel)
c                    pause

                     If(iskip .eq. 1)Then !only do symbass one time, skip after 1st time
      goto 798
                     Else
c                    print *,'chk 1 before copy_int'
c                    pause
      call copy_int(ncoef1,itempo1,ja)
     Endif
798  CONTINUE !dummy line for abv "If" statement
c                    print *,'passed 1st copy_int'
c                    print *,'ja(_) after copy_int in syseqn=',(ja(ic),ic=1,ncoef1)
c                    print *,'##########################'
c                    pause
c                    Endif
c788   continue !duumy line for skip=1
c                    print *,'ncoef1=',ncoef1
c                    stop

c                    If(getforce .eq. 1)then !skip, ja is already=ja2
c                    goto 112
c                    Else

c                    print *,'ncoef1=',ncoef1
c                    allocate ja2(ncoef1)
c     do 111 i=1,ncoef1
c                    print *,'ja(i)=',ja(i)
c111                 ja2(i)=ja(i)

c                    Endif
c                    print *,'ja2 just after setting ja=',(ja2(ic),ic=1,10)
c                    print *,'ja2(1) =',ja2(1)
c                    pause
c                    print *,''''''''''''''''''''''''''''''''''''''''
c                    pause
c112   Continue !dummy

c                    print *,'ja2 after set to ja =',(ja2(i),i=1,ncoef1)
c                    pause
c                    stop
      deallocate (itempo1)
c
c......sparse "numerical" assembly
c                    print *,'ianal=',ianal
c                    print *,'lumpmass=',lumpmass
     if (ianal .eq. 1) then
cc        allocate ( evalues(neig) )
cc        allocate ( evectors(ndof*neig) )
cc        allocate ( dm(ndof) )
cc        allocate ( elm(maxdofpe,maxdofpe) )
       if (lumpmass .eq. 0) then
cc        allocate ( am(ncoef1) )
       endif
     endif
c
c     print *, 'check point #01'
c     write(6,*) 'check point #01'
c                    print *,'ncoef1 before assembly=',ncoef1
c                    stop

c                    if(it .eq. 2)then
c                    print *,'it =2'
c                    print *,'before assembly2'
c                    stop
c                    endif
32   continue
c32   print *,'dummy line' !dummy line from "if(getforce .eq. 1) above
cvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
c     write(6,*) '*** passed symbass ***'
c                    If(getforce .eq. 1)then
c                    print *,'getforce =1 before assembly'
c                    print *,'ja2(i)=',(ja2(ic),ic=1,10)
c                    endif
c                    pause
c                    print *,'getforce=',getforce
                     If(getforce .eq. 1)then
c                    stop
c                    pause
c                    print *,'ncoef1=',ncoef1
c                    write(6,*)'ja(_) before assembly2=',(ja(i),i=1,ncoef1)
c                    print *,'ja(_) before assembly2=',(ja(i),i=1,ncoef1)
c                    stop
c                     call assembly2(ip,ad,an,lm,ie,je,propmat
c    $,        propsect,elk,be,b,iboundc,ux,uy
c    $,        ae,ia2,ja,x,y,ncoef1,maxdofpe,
c    $ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,tempo1,disp,
c    $iarc,ityp,getforce,it)
c                     print *,'ia2 before assembly=',(ia2(i),i=1,ndof+1)
c                     stop
c                     If(getforce .eq. 1)then
c                     print *,'ja2 before assembly & getforce=1',(ja2(ic),ic=1,10)
c                     endif
c                    pause

c                    print *,'ad stiff before assembly=',(ad(i),i=1,maxdofpe**2)
c                    print *,'ae stiff before assembly=',(ae(i),i=1,maxdofpe**2)
c                    pause
c                    print *,'skip call assembly'
c                    print *,'Before assembly 1 in sys'
c                    pause
     call assembly(ip,ad,an,lm,ie,je,propmat,
    $        propsect,be,b,iboundc,iarea,
    $        ae,ia,ja,x,y,z,ncoef1,maxdofpe,
c   $          ae,ia2,ja,x,y,z,ncoef1,maxdofpe,
    $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c    $ tempo1,disp,iarc,it,area,youngm,esm,qk,itnrmlon)
    $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon,istep,icheck,
    $          strainel,strainE,strainD,weight,Anew,Apart,Fin1,
    $          Fin2,VED,VEDold,xlic,SED)

c                     print *,'ia2 after assembly=',(ia2(i),i=1,ndof+1)
c                     print *,'*** Passed assembly ***'

c                    print *,'ad stiff after assembly=',(ad(i),i=1,maxdofpe**2)
c                    print *,'ae stiff after assembly=',(ae(i),i=1,maxdofpe**2)
c                    pause

c     STOP

                      else
c                    print *,'chk before assembly'
c                    print *,'getstiff=',getstiff
c                    print *,'getforce=',getforce
c                    print *,'Check before assembly'
c                    stop
c                    pause

c                                     call assembly(ip,ad,an,lm,ie,je,propmat
c    $,        propsect,elk,be,b,iboundc,ux,uy
c    $,        ae,ia2,ja,x,y,ncoef1,maxdofpe,
c    $ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,tempo1,disp,
c    $iarc,ityp,getforce,it)

c                     print *,'**ia2 before assembly**=',(ia2(i),i=1,ndof+1)
c                     stop
c                     print *,'going to assembly 2'
c                     print *,'ndofpe=',ndofpe
c                     print *,'chk before assembly'
c                     pause
c                    print *,'B(_) before assembly=',(b(ic),ic=1,ndof)
c                    pause
     call assembly(ip,ad,an,lm,ie,je,propmat,
    $        propsect,be,b,iboundc,iarea,
    $        ae,ia,ja,x,y,z,ncoef1,maxdofpe,
c   $          ae,ia2,ja,x,y,z,ncoef1,maxdofpe,
    $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
c    $ tempo1,disp,iarc,it,area,youngm,esm,qk,itnrmlon)
    $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon,istep,icheck,
    $          strainel,strainE,strainD,weight,Anew,Apart,Fin1,
    $          Fin2,VED,VEDold,xlic,SED)

c                     print *,'***Passed assembly***'
c                    print *,'maxdofpe=',maxdofpe
c                    print *,'ad stiff after assembly=',(ad(ic),ic=1,maxdofpe**3)
c                    print *,'an stiff after assembly=',(an(ic),ic=1,50)
c                    print *,'ae stiff after assembly=',(ae(i),i=1,maxdofpe**2)
c                    pause
c                    stop
c                    write(6,*)'***Passed assembly***'

c                    stop
                     endif


c----------------------------------------------------
c... Only used when solving, getsolve=1
c                    print *,'getforce=?',getforce
c                    print *,'getstiff=?',getstiff
c                    print *,'getsolve=?',getsolve
c                    pause
c                    print *,'B(_)=',(b(ic),ic=1,ndof)
c                    pause
1111                 If(getsolve .eq. 1)Then
                     goto 797
cx     call assembly(ip,ad,an,lm,ie,je,propmat,
cx    $        propsect,be,b,iboundc,
cx    $        ae,ia,ja,x,y,z,ncoef1,maxdofpe,
cx    $ ndof,neltype,nel,ndofpe,ianal,lumpmass,dm,am,elm,
cx    $ tempo1,disp,iarc,it,area,youngm,qk,itnrmlon)
c----------------------------------------------------
c                    print *,'b after assembly getsolve=1 =',(b(i),i=1,ndof)
c                    print *,'ad after assembly getsolve=1',(ad(i),i=1,maxdofpe**2)
c                    print *,'ae after assembly getsolve=1',(ae(i),i=1,maxdofpe**2)
c                    stop
c                    pause
                     Endif


c                    write(6,*)'ae=',(ae(i),i=1,maxdofpe*2)
c                    write(6,*)'disp=',(disp(i),i=1,ndof)
c                    write(6,*)'ae=',(ae(i),i=1,20)
c                    write(6,*)'disp=',(disp(i),i=1,10)
c                    stop
```

```
c                 print *,'Getforce after assembly2 = ',getforce
c                 stop
                  If(getforce .eq. 1)then

c                 print *,'b after assembly getforce is 1 =',(b(i),i=1,ndof)
c                 print *,'elk after assembly getforce is 1=',(elk(i),i=1,ndof)
c                 print *,'iboundc=',(iboundc(i),i=1,ndof)
c***need to get support dof into "b"
                  Do 449 1=1,ndof
                  If(iboundc(i) .eq. 1)Then
                  b(i)=0.0
                  Endif
449   Continue
c                 b(1)=0.0
c                 b(2)=0.0
c                 b(3)=0.0
c                 b(4)=0.0
c                 b(6)=0.0
c                 b(7)=0.0
c                 b(8)=0.0
c                 b(9)=0.0
c                 print *,'b modify for support dof =',(b(i),i=1,ndof)
c                 pause
                   goto 45 !this goes to the end of subroutine when we want internal
forces
c                 stop
                  endif
c**********
c                  print *,'ae = ',(ae(i),i=1,maxdofpe**2)
c                  stop

c     write(6,*) 'check point #02'
c                 print *,'check point #02'
c                 stop
cc     if (ianal .eq. 1) then
cc     write(6,*) 'diag-mass = dm(-) = ',(dm(i),i=1,ndof)
cc     endif
cc     if (ianal .eq. 1  .and.  lumpmass .eq. 0) then
cc     write(6,*) 'ofldiag-mass = am(-) = ',(am(i),i=1,ncoef1)
cc     endif
c
c.....assuming to skip the "REORDERING" process (minimize "fills-in" terms)
c
c                 print *,'**before minfills***'
c                 print *,'iperm before minfills=',(iperm(i),i=1,ndof)
c                 pause
c                 stop

c                 print *,'iskip inside syseqn=',iskip
c                 pause

                  If(iskip .eq. 1)Then !only do symbass one time, skip after 1st time
c                 print *,'iskip=1 goto 797'
c                 pause
      goto 797
                  Else
     call minfills(ndof,iperm)
c                 print *,'iperm after minfills=',(iperm(i),i=1,ndof)
c                 pause
c_____________________________________________________________________
                   iskip = 1!This causes symbass to be skipped, only need to go once
c           save for other iterations!!!
c_____________________________________________________________________
c                 print *,'passed minfills'
c                 print *,'iperm after minfills=',(iperm(i),i=1,ndof)
                  Endif
797   Continue !dummy statement
c                 stop
c.....eigen-solution
cc     if (ianal .eq. 1) then
cc     lump=lumpmass
cc     ishift=0
cc     iprint=1
cc     mtot=20000000
cc     allocate( tempo3(mtot) )
c
c     r(mtot)=working array (in real*8) = tempo3(-)
c     iflag(1,2,3,4,5,6,7,8,9,10)=nreord,loop_unroll,??,??,neig
c                    =iprint,lump,ishift,ncoef2,??
c     nreord=0,1,2,3=no reorder,metis,nd,mmd
c
cc      iflag(1)=0          ! 1,2,3 = metis,nd,mmd
cc      iflag(2)=1          ! level of unroll
cc      iflag(5)=neig
cc      iflag(6)=iprint
cc      iflag(7)=lump
cc      iflag(8)=ishift
c     iflag(9)=ncoef2     ! as an output
c
cc      write(6,*) 'inside *cmei*.f: ndof,ncoef1,neig,lump = '
cc      write(6,*)                   ndof,ncoef1,neig,lump
cc      write(6,*) 'inside *cmei*.f: ishift,mtot = '
cc      write(6,*)                   ishift,mtot
cc      write(6,*) 'inside *cmei*.f: iflag(10) = ',(iflag(i),i=1,10)
cc      write(6,*) 'inside *cmei*.f: ia(-) = ',(ia(i),i=1,ndof+1)
cc      write(6,*) 'inside *cmei*.f: ja(-) = ',(ja(i),i=1,ncoef1)
cc      write(6,*) 'inside *cmei*.f: ad(-) = ',(ad(i),i=1,ndof)
cc      write(6,*) 'inside *cmei*.f: an(-) = ',(an(i),i=1,ncoef1)
cc      write(6,*) 'inside *cmei*.f: dm(-) = ',(dm(i),i=1,ndof)
c
ccc     if (lump .eq. 1) then
ccc     call eigsolver011(ndof,ncoef1,neig,lump,ishift,
ccc    $mtot,ia,ja,ad,an,dm,   evalues,evectors,tempo3,iflag,iperm)
ccc     elseif (lump .eq. 0) then
ccc     call eigsolver022(ndof,ncoef1,neig,lump,ishift,
ccc    $mtot,ia,ja,ad,an,dm,am,evalues,evectors,tempo3,iflag,iperm)
ccc     endif
cc      deallocate( tempo3 )
cc      STOP
cc      endif

                  If(getstiff.eq.1)Then
                  goto 674
                  Endif

c.....sparse "symbolic" factorization
      ncoef2_guess=1000
c                 print *,'^^^^^^^Chk before allocate itempo1^^^^^^'
c                 pause
      allocate ( itempo1(ncoef2_guess) )
c                 print *,'ia before symfactd=',(ia(ic),ic=1,12)
c                 print *,'ja before symfactd=',(ja(ic),ic=1,3)
c                 print *,'++++ Chk before symfactd ++++'
c                 print *,'getstiff=',getstiff
c                 print *,'getforce=',getforce
c                 print *,'getsolve=',getsolve
c                 pause
c                 print *,'ndof before symfactd=',ndof
c                 pause
      call symfactd(ndof,ia,ja,iu,itempo1,ip,ncoef2)
c     call symfactd(ndof,ia2,ja,iu,itempo1,ip,ncoef2)

c                 print *,'***************passed symfactd************'
c     print *, 'after symfactd: IU(-) = ',(iu(ic),ic=1,ndof+1)
c                 pause
c                 pause

c                 write(6,*)'passed symfactd'
c                 stop
c     write(6,*) 'check point #03'
c
c                  print *,'ncoef2=',ncoef2
c                 print *,'NNNNNNNNNNNN ncoef2 NNNNNNNNNNNN'
c                 pause
c                 stop
      allocate ( ju(ncoef2) )
      allocate ( jut(ncoef2) )
      allocate ( un(ncoef2) )
c                 print *,'itempo1 before copy_int=',(itempo1(ic),ic=1,28)
c                 pause
c                 print *,'chk 2 before copy_int'
       call copy_int(ncoef2,itempo1,ju)
c     subroutine copy_int(n,iarray,icopy)

c                 print *,'passed copy_int'
c                 print *,'getsolve=1',getsolve
c                 print *,'itempo1 after copy_int=',(itempo1(ic),ic=1,28)
c                 print *,'ju after copy_int=',(ju(ic),ic=1,28)

c                 pause
c                 pausc

c     print *,'***Passed copy_int***'
c                 print *,'ad stiff after copy_int=',(ad(i),i=1,maxdofpe**2)
c                 print *,'an off-stiff after copy_int=',(an(ic),ic=1,3)
c                 print *,'ae stiff after copy_int=',(ae(i),i=1,maxdofpe**2)
c                 pause
c                 print *,'getsolve=',getsolve
                  If(getsolve.eq.1)then !skip, deallocate
                   allocate (tempo1(10000) )
      deallocate (itempo1)
c                 print *,'chk after allocate'
                  goto 781
                  Endif
      deallocate (itempo1)
781   continue
c                 print *,'Chk after 781'
c
c.....transpose twice to put column numbers in order
      call transad(ndof,ndof,iu,ju,iut,jut)
c     subroutine transad(n,m,ia,ja,iat,jat)
c                 print *,'*** passed transa  ***'
c     write(6,*) '*** passed transa ***'
c                 print *,'iu after transad=',(iu(i),i=1,ndof+1)
c                 write(6,*)'iu after transad=',(iu(i),i=1,ndof+1)
c                 pause
c                 stop
c
c.....assuming to skip the "SUPER NODE" process (for unrolling purpose)
c
      call supernode(ndof,isupn)
c                 print *,'*** passed supernode  ***'
c                 stop
c
c.....sparse "numerical" factorization
c                 print *,'ia2 before numfa1d= ',(ia2(i),i=1,ndof+1)
c                 write(6,*)'ia2 before numfa1d= ',(ia2(i),i=1,ndof+1)
c                 stop
c                  print *,'ndof before numfa1d=',ndof
c                  pause

c                 print *,'getstiff before numfa1d=',getstiff
c                 print *,'chk before numfa1c'
c                 pause
674               If(getstiff .eq. 1)Then !skip, only need stiff
                  Goto 793
                  Endif

      call numfa1d(ndof,ia,ja,ad,an,iu,
     $           ju,di,un,ip,iup,
     $           isupn,iopf,istep)
c routine numfa1d(n,ia,ja,ad,an,iu,ju,di,un,ip,iup,isupd,iopf)
c     call numfa1d(ndof,ia2,ja,ad,an,iu,
c    $           ju,di,un,ip,iup,
c    $           isupn,iopf)

793   Continue !dummy line
c                 print *,'getstiff after numfa1d=',getstiff
c                 pause

                  If(getstiff .eq. 1)Then !skip, only need stiff
                  Goto 675
                  Endif
```

```
c                    print *,'*** PASSED NUMFA1d  ***'
c                    pause
c                    STOP
c     write(6,*) '*** passed numfa1d ***'
c.....sparse forward/backward solution phase

c                    write(6,*)'ndof before fbcd=',ndof
c                    write(6,*)'b before fbcd=',(b(i),i=1,ndof)
c                    write(6,*)'di before fbcd=',(di(i),i=1,ndof)
c                    write(6,*)'iu before fbcd=',(iu(i),i=1,ndof+1)
c                    write(6,*)'ju before fbcd=',(ju(i),i=1,ncoef2)
c                    write(6,*)'un before fbcd=',(un(i),i=1,ncoef2)
c                    print *,'ndof before fbcd=',ndof
c                    print *,'b before fbcd=',(b(i),i=1,ndof)
c                    print *,'di before fbcd=',(di(i),i=1,ndof)
c                    print *,'iu before fbcd=',(iu(i),i=1,ndof+1)
c                    print *,'ncoef2=',ncoef2
c                    print *,'ju before fbcd=',(ju(ic),ic=1,ncoef2)
c                    pause
c                    print *,'un before fbcd=',(un(i),i=1,ncoef2)
c                    stop

c    print *,'////////////////////////////////'
c                    print *,'disp @ before fbcd =',(disp(i),i=1,neq)
c                    pausc
c                    print *,'////////////////////////////////'
c                    pause
c                    print *,'going to fbcd'
c                    pause
      call fbcd(ndof,iu,ju,di,un,b,
     $         tempo1,iopb,isupn,getsolve)

c...tempo1 are displacements!
c                    print *,'tempo1=',(tempo1(i),i=1,neq)
c                    print *,'***  passed fbcd  ***'
c                    pause

c                    write(6,*)'*** passed fbcd ***'
c*****************************
c                     deallocate (ja)
c                     deallocate (ju)
c*****************************
c                    stop

                     do 40 i=1,ndof
c                    qk(i)=tempo1(i)
40                   disp(i)=tempo1(i)
c                    print *,'tempo1, disp = ',tempo1
c                    print *,'b = ',(b(i),i=1,ndof)
c                    print *,'b after assembly getforce is 1 =',(b(i),i=1,20)
c***need to get support dof into "b"
c                    b(8)=0.0
c                    b(9)=0.0
c                    b(10)=0.0
c                    write(6,*)'b end of syseqn = ',(b(i),i=1,ndof)
c                    print *,'b end of syseqn = ',(b(i),i=1,20)
c                    stop
c    print *,'////////////////////////////////'
c                    print *,'disp @ end of syseqn =',(disp(i),i=1,12)
c                    pause
c                    print *,'////////////////////////////////'
c                    pause
c                    pause
c                    pause
c                    pause
c                    pause
c                    stop
                     If(getforce .eq. 1)then
c                    print *,'disp end syseqn&getforce is 1 =',(disp(i),i=1,ndof)
c                    stop
                     endif
675  continue !blank line used when getstiff=1
45                   continue !blank line used to go if only getting internal forces
c793  continue! dummy,used to go if only get stiffness

                     return
                     end
C*********************************************************************
*
C***              SUBROUTINE: RESID                          ***
C*********************************************************************
*
     subroutine resid(dqtot,dqi,dqim1,qk,b,exfor)
     implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
     common/RIK/ stifi(60,60),dqk1(5000),
    $     dqk2(5000),fintk(5000),
    $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c        common/deltaqi/dqi(500)

     common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
     common/testing/toldis,tolfor,exlk,iconv

     dimension dqtot(*),dqim1(*),b(*),exfor(*),qk(*)

c     dimension exfor(neq),fintk(neq),resk(neq)
c
ccc      print *,'a inside resid=',a
c     print *,'exlk inside resid=',exlk
c     print *,'exfor inside resid=',(exfor(i),i=1,neq)
c     print *,'b inside resid=',(b(i),i=1,neq)

c     print *,'neq inside resid=',neq
c     stop
     do 10 i=1,neq
ccc         resk(i) = exlk*cxfor(i) - a(neq+i) !!!For 2D truss only
          resk(i) = exlk*exfor(i) - b(i)
c       print *,'i=',i,'exlk inside resid =',exlk
c       resk(i) = exlk*exfor(i) - fintk(i)
 10   continue
ccc     print *,'exlk inside resid = ',exlk
ccc     print *,'exfor(i) inside resid =',exfor
c     print *,'resk(i) inside resid = ',resk
c     print *,'exlk inside resid = ',exlk
c     print *,'exfor(i) inside resid =',(exfor(i),i=1,neq)
c     print *,'resk(i) inside resid = ',(resk(i),i=1,neq)
c                    pause

c     stop

c     stop
c
     return
     end
c*********************************************************************
c***              SUBROUTINE: TEST                           ***
c*********************************************************************
c     subroutine test(toldis,tolfor,exlk,iconv)
     subroutine test(dqtot,dqim1,qk,exfor,b)
     implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
     common/RIK/ stifi(60,60),dqk1(5000),
    $     dqk2(5000),fintk(5000),
    $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c        common/deltaqi/dqi(500)

     common/testing/toldis,tolfor,exlk,iconv

     dimension dqtot(*),dqim1(*),exfor(*),qk(*),b(*)

                    dimension dqk(5000)

c     dimension dqk(neq),qk(neq),exfor(neq),resk(neq),scali(neq)
c
c     print *,'exlk inside TEST=',exlk
c     print *,'tolfor inside TEST=',tolfor
c                    print *,'dqtot inside TEST=',(dqtot(i),i=1,neq)
c                    pausc
c     print *,'qk(i) insidc resid =',(qk(i),i=1,neq)
c                    pause
c     stop
     iconv = 0
c
     if (tolfor .lt. 1.d0) then
c     print *,'tolfor = ',tolfor
c     print *,'goto unbalf'
cc     print *,'tolfor, exlk, iconv inside test = ',tolfor,exlk,iconv
c  c     pause
c       stop
c       call unbalf(tolfor,exlk,iconv)
c     print *,'exlk before unbalf = ',exlk
c                    stop
c  c     pause
c     print *,'chk before unbalf in test'
c                    pausc
       call unbalf(dqtot,dqim1,qk,exfor,b)
     end if

c     print *,'toldis after unbalf = ',toldis

c     stop
c
     if (toldis .lt. 1.d0) then
c     print *,'toldis = ',toldis
c     print *,'goto displ'
c     STOP
c       call displ(toldis,iconv)
       call displ(dqtot,dqk,dqim1,qk,b)
     end if
c
     return
     end
c*********************************************************************
c***            SUBROUTINE: UNBALF                           ***
c*********************************************************************
c     subroutine unbalf(tolfor,exlk,iconv)
     subroutine unbalf(dqtot,dqim1,qk,exfor,b)
     implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
     common/RIK/ stifi(60,60),dqk1(5000),
    $     dqk2(5000),fintk(5000),
    $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c        common/deltaqi/dqi(500)

     common/testing/toldis,tolfor,exlk,iconv

     dimension dqtot(*),dqim1(*),exfor(*),qk(*),b(*)

c     dimension exfor(neq),resk(neq),scali(neq)
c
c     print *,'Start of "UNBALF"'
c     print *,'exlk inside UNBALF=',exlk
c     stop
     unbfi = 0.d0
     unbfc = 0.d0
c
```

```
c ... compute numerator of eqn. 4.133
c ... compute denominator of eqn. 4.133
c     print *,'scali(i) inside UNBALF'=',(scali(i),i=1,neq)
c     print *,'resk(i) inside UNBALF'=',(resk(i),i=1,neq)
c                     pause
c     print *,'exfor(i) inside UNBALF'=',(exfor(i),i=1,neq)
c       stop
      do 10 i=1,neq
        unbfi = unbfi + dabs(scali(i))*resk(i)**2
        unbfe = unbfe + dabs(scali(i))*exfor(i)**2
 10   continue
c
c ... compute eqn. 4.133
c     print *,'unbfi=',unbfi
c     print *,'unbfe=',unbfe
c                     print *,'scali(_)=',(scali(ic),ic=1,neq)
c     print *,'exlk=',exlk
      delf = dsqrt(unbfi)/dsqrt(unbfe)/exlk
c     print *,'delf inside unbal = ',delf
c                     pause
C     stop
c
c ... test to see if delf is greater than tolfor
      if (dabs(delf) .gt. tolfor) then
        iconv = iconv + 100
      end if
c     print *,'iconv inside unbal = ',iconv
c     stop
c
      return
      end
c *********************************************************************
c ***             SUBROUTINE: DISPL                                 ***
c *********************************************************************
c     subroutine displ(toldis,iconv)
      subroutine displ(dqtot,dqi,dqim1,qk,b)
      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(60,60),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

c         common/deltaqi/dqi(500)

      common/testing/toldis,tolfor,exlk,iconv
      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2

                      dimension dqtot(*),dqim1(*),qk(*),b(*)

                      dimension dqk(5000)

c     dimension dqk(neq),qk(neq)
c
c     print *,'neq inside displ=',neq
c       stop
      dqsq = 0.d0
      qsq = 0.d0
c
c ... compute deld using eqn. 4.132
      do 10 i=1,neq
c       dqsq = dqsq + dqk(i)**2
        dqsq = dqsq + dqk(i)**2
c         qsq = qsq + qk(i)**2
        qsq = qsq + b(i)**2
c       qsq = qsq + a(i)**2
 10   continue
c     print *,'dqsq inside displ =',dqsq
c     print *,'qsq inside displ =',qsq
c       stop
      deld = dsqrt(dqsq)/dsqrt(qsq)
c     print *,'deld inside displ = ',deld
c                     PAUSE
c     stop
c
c ... test to see if deld is greater than toldis
      if (deld .gt. toldis) then
        iconv = iconv + 10
      end if
c     print *,'iconv inside displ = ',iconv
c     stop
c
      return
      end
C *********************************************************************
C ***             SUBROUTINE: RESULT                                ***
C *********************************************************************
c     subroutine result(istep,it,dqtot,dqi,dqim1,qk,exfor,ad,
c    $                  strainel,strainE,strainD)
      subroutine result(istep,it,dqtot,dqim1,qk,exfor,propsect,ad,
     $                  strainel,strainE,strainD,weight,ncl,Anew,Fin1,
     $                  xlie,weightotal,Nevenodd,N)

c         call result(istep,it,dqtot,dqi,dqim1,qk,exfor,ad,
c    $                  straincl,strainE,strainD)

      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c    $          resk(50),scali(50),work1(50),indx(50),neq
c     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
c    $          dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c     common/RIK/ stifi(60,60),dqk1(5000),
c    $     dqk2(5000),
c    $     resk(5000),scali(5000),work1(5000),indx(5000),neq
      common/RIK/ stifi(60,60),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq
c     common/deltaqi/dqi(500)

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
      common/testing/toldis,tolfor,exlk,iconv

c                     dimension dqtot(*),dqim1(*),qk(*),ad(*)
                      dimension dqtot(*),dqim1(*),exfor(*),qk(*),ad(*)

                      dimension dqi(5000),dqk(5000)

                      dimension strainel(*),strainE(*),strainD(*),weight(*)

                      dimension propsect(*) !Area of members

                      dimension SEDratio(100)

                      dimension Anew(*),Fin1(*),xlie(*)

ccc     common/disp/ux(2),uy(2),disp(500)

cccc      print *,'ux(1) & ux(2) =',ux(1),ux(2)
c     print *,'stop in result'
c     stop
c     dimension exfor(neq),qk(neq)
c
c     if (istep .eq. 1) then
c  c     write(*,10)
c  c     write(*,20)
c       write(11,10)
c       write(11,20)
c     end if
c
      open(15,file='fea3d2_out.txt',status='unknown')
                      open(21,file='Stiffout.txt',status='unknown')
c     open(16,file='strainD.txt',status='new')
      open(17,file='strainE.txt',status='unknown')
      open(18,file='strainD.txt',status='unknown')
c                     open(19,file='resk.txt',status='unknown')
                      open(19,file='Fin.txt',status='unknown')

                      open(22,file='SEDratio.txt',status='unknown')
                      open(25,file='AreaNew.txt',status='unknown')

c                     If(Nevenodd.eq.2)then !N is now even number so print!
c                     If(N.eq.3)then

c                     print *,'Nevenodd=',Nevenodd

c                     open(20,file='displacement.txt',status='unknown')
c     write(*,50) '*',istep,exlk,qk(1),qk(2),it

c                     print *,'********** Inside Result **********'
c                     print *,'istep=',istep
c                     pause

c     Do 51 i=1,neq
c     xinternalforce = exlk * exfor(3)
c     yinternalforce = exlk * exfor(4)
caa     appliedforce = exlk * exfor(5)
carch     appliedforce = exlk * exfor(62) !arch
c                     print *,'exlk=',exlk
c                     print *,'exfor(62)=',exfor(62)
c                     pausc
c     appliedforce = exlk * exfor(6)

c     aforce=a(12)
c     appliedforce = exlk * exfor(15)
c     appliedforce = exlk * exfor(3)
c18     appliedforce = exlk * exfor(20)
carch     appliedforce = exlk * exfor(44)
cx     print *,'exlk inside result = ',exlk
cccc     print *,'exfor(12) inside result = ',exfor(12)
cc     print *,'exfor(15) inside result = ',exfor(15)
c     print *,'exfor(3) inside result = ',exfor(3)
c     print *,'appliedforce inside result=',appliedforce
cc     print *,'a(4) displacement = ',a(4)
cc     print *,'a(3) displacement = ',a(3)
c       pausc
c       stop
cccc      write(*,50) '*',istep,exlk,xinternalforce,yinternalforce,
cccc     c                a(3),a(4),it
ccc     write(*,50) '*',istep,exlk,xinternalforce,yinternalforce,
ccc     c                a(3),a(4),it
c     write(*,50) '*',istep,exlk,appliedforce,
c     c                a(3),a(4),it

c       write(12,55) appliedforce
c 55    format(f12.4)
c       write(14,56) a(6)
c 56    format(f12.4)
c     write(15,50) '*',istep,exlk,appliedforce,a(3),it
c 50  format(a1,1x,i3,1x,f10.6,1x,f10.6,1x,f10.6,1x,f10.6,1x,f10.6,
c     c          1x,i3)
c 50  format(a1,1x,i3,1x,f12.4,1x,f12.4,1x,f12.4,1x,f12.4,1x,
c     c       f12.4,1x,f12.4,1x,i3)
c below for T3
c 50  format(a1,1x,i3,1x,f12.4,1x,f12.4,1x,f12.4,1x,f12.4,1x,
c     c       f12.4,1x,f12.4,1x,i3,1x,f12.4)
c below for T3x
ccc 50  format(a1,1x,i5,1x,f12.4,1x,f16.0,1x,f12.4,1x,f12.4,1x,
ccc     c       f12.4,1x,f12.4,1x,f12.4,1x,f12.4,2x,i5)
c 50  format(a1,1x,i3,1x,f10.6,1x,f12.4,1x,f10.6,1x,f10.6,1x,
c     c       f10.6,1x,f10.6,1x,f10.6,1x,f10.6,1x,i3)
c 50  format(a1,1x,i3,1x,f10.6,1x,f10.6,1x,f10.6,1x,f10.6,
c     c          1x,i3)
c     print *,'check inside result'

c     if(istep .eq. 1) then
c     write(12,55)
c 55  format('Appliedforce')
c     write(13,56)
```

```
c 56  format('Two ele Truss')

carch     write(15,53)
carch 53  format(' istep    exlk      appliedforce     dqtot(61)
carch    $ dqtot(62)     it')

c 53  format(' istep    exlk      appliedforce     dqtot(4)
c    $ dqtot(5)     it')

c 53  format(' istep    exlk   appliedforce   a(3)    a(4)
c    $a(5)    a(6)   it')
c 53  format(' istep    exlk      appliedforce        a(1)
c    $    a(2)      a(3)    a(4)    a(5)     a(6)     it')

c     write(*,53)
c 53  format(' istep    exlk   appliedforce   a(3)    a(4)
c    $a(5)    a(6)    a(9)    a(10)    it')
c 53  format('  istep    exlk     appliedforce      a(5)      a(6)
c    $       a(13)      a(14)      a(11)      a(12)   it')
c18 53  format('  istep    exlk     appliedforce      a(15)
c18    $ a(16)      a(17)      a(18)      a(19)      a(20)  it')
CCC 53  format('  istep    exlk     appliedforce      a(19)
CCC    $ a(20)      a(41)      a(42)      a(43)      a(44)  it')
carch 53  format('  istep    exlk     appliedforce      a(47)
carch    $ a(48)      a(39)      a(40)      a(43)      a(44)  it')
ccc 53  format('  istep    exlk     appliedforce      a(10)
ccc    $ a(11)      a(12)   it')
c 53  format('  istep    exlk     appliedforce      a(13)
c    $ a(14)      a(15)   it')
cc 53  format('  istep    exlk     appliedforce      a(1)
cc    $  a(2)      a(3)    it')
cdome 53  format('  istep    exlk     appliedforce      a(6)
cdome    $ a(9)      a(3)   it')
c 53  format(' istep    exlk   appliedforce   a(3)    a(4)
c    $a(5)    a(6)   it')
cc below for T3
cc 53  format(' istep    exlk    appliedforce     a(3)      a(4)
cc    $  a(5)      a(6)      it      a(12)')
c below for T3x
cxxx 53  format(' istep    exlk   appliedforce     a(3)      a(4)
cxxx    $  a(5)      a(6)      a(9)      a(10)     it')
c below for T3y
c 53  format(' istep    exlk   appliedforce      a(1)      a(2)
c    $  a(3)      a(4)      it      a(12)')
c 53  format(' istep    exlk    xinfor    yinfor    a(3)     a(4)
c    $   it')
c     endif

c     print *,'check 2 inside result'
c     write(15,50) '*',istep,exlk,appliedforce,
c     c          a(3),a(4),a(5),a(6),a(11),a(12),it
c Below IS FOR T3
c     write(15,50) '*',istep,exlk,appliedforce,
c     c          a(3),a(4),a(5),a(6),it,a(12)
c     write(15,50) '*',istep,exlk,appliedforce,
c     c          a(4),a(5),a(6),it
c6    write(15,50) '*',istep,exlk,appliedforce,
c6    c          a(1),a(2),a(3),a(4),a(5),a(6),it

c Below IS FOR T3x
c     write(15,50) '*',istep,exlk,appliedforce,
c     c          a(5),a(6),a(13),a(14),a(11),a(12),it
c18    write(15,50) '*',istep,exlk,appliedforce,
c18    c          a(15),a(16),a(17),a(18),a(19),a(20),it
c18 50  format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,f10.4,
c18    c          f10.4,6x,  f10.4,  f10.4,6x,i3)
CCC     write(15,50) '*',istep,exlk,appliedforce,
CCC    c          a(19),a(20),a(41),a(42),a(43),a(44),it
carch    write(15,50) '*',istep,exlk,appliedforce,
carch    c          a(39),a(40),a(47),a(48),a(43),a(44),it
carch 50  format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,3X,f10.4,
carch    c          f10.4,3x,  f10.4,3X f10.4,4x,i5)
cc     print *,istep,exlk,appliedforce,
cc    c          a(10),a(11),a(12),it
ccc     write(15,50) '*',istep,exlk,appliedforce,
ccc    c          a(10),a(11),a(12),it

c 50  format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,f10.4,
c     c          6x,i3)
c6 50  format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,4x,f10.4,
c6    c          f10.4,f10.4,f10.4,4x,i3)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c          a(13),a(14),a(15),it
cc     write(15,50) '*',istep,exlk,appliedforce,
cc    c          a(1),a(2),a(3),it
caa     write(15,50) '*',istep,exlk,appliedforce,
caa    c          dqtot(4),dqtot(5),it
c**********************************************************************
c** "cris_sdof_in.dat"*************************************************
c** "Khot_2ele_mod_in.dat"*********************
cx                  weightotal=0.d0
cx       appliedforce = exlk * exfor(3)
c       appliedforce = exlk * exfor(5)
c       appliedforce = exlk * exfor(15)
cx                  Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
cx                  weightotal=weightotal+weight(i)
cx55                Continue
c                   print *,'weightotal final=',weightotal
c                   pause

cx     if(istep .eq. 1) then
c     write(15,53)
c 53  format('  istep    exlk      appliedforce      a(1)
c    $    a(2)      a(5)     it      ad(3)     ad(4)')

cx     write(15,53)
cx 53  format('  istep    exlk      appliedforce      a(1)
cx    $  a(2)      a(3)     it    Weight    Area')

c 53  format('  istep   exlk      appliedforce  a(13)
c    $ a(14)      a(15)    it    Weight    Area')


c     write(15,53)
c 53  format('  istep    exlk      appliedforce      a(5)
c    $   Strain    S_energy   SE_density      it    resk(4)
c    $   resk(5)')

c                   write(16,62)
c62    format('STRAIN')
c                   write(17,63)
c63    format('STRAIN ENERGY')
c                   write(18,64)
c64                 format('STRAIN ENERGY DENSITY')
c                   write(19,65)
c65    format('Appliedforce')
c                   write(20,66)
c66    format('Displacement')

cx     endif

c     write(15,50) '*',istep,exlk,appliedforce,
c     c    dqtot(4),dqtot(5),it,weightotal,propsect(1)
c50   format(a1,2x,i4,5x,e14.6,6x,e10.4,3X,e10.4,3x,e10.4,
c     c          4x,i5,3x,e10.4,3x,e10.4)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c    dqtot(1),dqtot(2),dqtot(5),it,
c     c    ad(1),ad(2)
c50   format(a1,2x,i4,5x,c14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
c     c          4x,i5,4x,c10.4,3x,c10.4)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c    dqtot(3),dqtot(4),dqtot(5),it,weightotal
c 50   format(a1,1x,i4,1x,e14.6,1x,e10.2,2x,e10.4,2x,e10.4,1X,e10.4,
c     c          1x,i5,1x,e10.4)

cx     write(15,50) '*',istep,exlk,appliedforce,
cx    c    dqtot(1),dqtot(2),dqtot(3),it,weightotal,propsect(1)
cx50   format(a1,1x,i4,1x,e14.6,1x,e10.2,2x,c10.4,2x,e10.4,1X,e10.4,
cx    c          1x,i5,1x,e10.4,1x,c10.4,1x,c6.4)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c    dqtot(13),dqtot(14),dqtot(15),it,weightotal,propsect(1)
c50   format(a1,1x,i4,1x,c14.6,1x,e10.2,2x,e10.4,2x,e10.4,1X,e10.4,
c     c          1x,i5,1x,c10.4,1x,e10.4,1x,e6.4)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c    dqtot(5),strainel(1),strainE(1),strainD(1),it,
c     c    resk(4),resk(5)
c50   format(a1,2x,i4,5x,e14.6,3x,c10.2,6x,e10.4,
c     c          3x,e12.6,3x, e12.6,3X c12.6,4x,i5,
c     c          2x,e12.6,2x,e12.6)

c                   write(16,72) strainel(1)
c72                 format(e12.6)
c                   write(17,72) strainE(1)
c                   write(18,72) strainD(1)
c     write(19,72) appliedforce
c     write(20,72) dqtot(5)

c                   If(Nevenodd.eq.2)then !N is now even number so print!
c                   If(N.eq.3)then

c** "Khot_2ele_mod_in.dat"*************************************************
      appliedforce = exlk * exfor(5)
      weightotal=0.d0
c                   Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
cd     SEDratio(i)=strainD(i)/weight(i)
                    weightotal=weightotal+weight(i)
55                  Continue

     if(istep .eq. 1) then
c                   write(15,63)'N =',N
c 63   format(i4)
     write(15,53)
 53   format('  istep    exlk      appliedforce      a(4)
     $  a(5)    it      Weight    Area1     Area2
     $       xlic 1      xlie 2    fin1   N  Nevenodd')
                    endif

c                   If(Nevenodd .eq. 2 .or. Nevenodd .eq. 0)then

     write(15,50) '*',istep,exlk,appliedforce,
     c   dqtot(4),dqtot(5),it,weightotal,
     c   propsert(1),propsect(2),xlie(1),xlic(2),fin1(1),N,
     c   Nevenodd

c                   endif

50   format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
     c 1x,i5,1x,c14.6,2x,c12.6,2x,e12.6,2x,e14.8,2X,e14.8,1x,e8.2,
     c i4,2x,i4)

c** "Khot_4ele_mod_in_initial.dat"*************************************************
cKhot4      appliedforce = exlk * exfor(15)
cKhot4      weightotal=0.d0
cKhot4              Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
```

```
cd     SEDratio(i)=strainD(i)/weight(i)
cKhot4              weightotal=weightotal+weight(i)
cKhot455            Continue

cKhot4      if(istep .eq. 1) then
cKhot4      write(15,53)
cKhot4 53   format('  istep     exlk       appliedforce       a(14)
cKhot4    $  a(15)    it       Weight     Area1        Area2
cKhot4    $  Area3        Area4     xlie 1       xlie 2     fin1')
cKhot4              endif

cKhot4      write(15,50) '*',istep,exlk,appliedforce,
cKhot4     c   dqtot(14),dqtot(15),it,weightotal,
cKhot4     c   propsect(1),propsect(2),propsect(3),propsect(4),
cKhot4     c   xlie(1),xlie(2),fin1(1)

cKhot450    format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
cKhot4     c 1x,i5,1x,e14.8,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
cKhot4     c  e14.8,2X,e14.8,1x,e8.2)


c**
"Dome2_opt_mod_in_initial.dat"*********************************************
****
cdome       appliedforce = exlk * cxfor(3)
cdome       weightotal=0.d0
cdome               Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
cd     SEDratio(i)=strainD(i)/weight(i)
cdome               weightotal=weightotal+weight(i)
cdome55             Continue

cdome       if(istep .eq. 1) then
cdome       write(15,53)
cdome 53    format('  istep     exlk       appliedforce       a(2)
cdome     $  a(3)       it     Weight      Area1        Area2
cdome     $  Area3        Area4      xlie 1       xlie 2       fin1')
cdome               endif

cdome       write(15,50) '*',istep,exlk,appliedforce,
cdome      c   dqtot(2),dqtot(3),it,weightotal,
cdome      c   propsect(1),propsect(2),propsect(3),propsect(4),
cdome      c   xlie(1),xlie(2),fin1(1)

cdome50    format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
cdome      c 1x,i5,1x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
cdome      c  e12.6,2X,e12.6,1x,e8.2)
c**************************************************************************

c**
"Arch3d_opt_mod_in_initial.dat"*********************************************
****
cArch3d       appliedforce = exlk * exfor(65)
cArch3d       weightotal=0.d0
cArch3d             Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
cd     SEDratio(i)=strainD(i)/weight(i)
cArch3d             weightotal=weightotal+weight(i)
cArch3d55           Continue

cArch3d      if(istep .eq. 1) then
cArch3d      write(15,53)
cArch3d 53   format('  istep     exlk       appliedforce       a(64)
cArch3d    $  a(65)       it     Weight      Area1        Area2
cArch3d    $  Area3        Area4      xlie 50       xlie 51       fin1')
cArch3d             endif

cArch3d      write(15,50) '*',istep,exlk,appliedforce,
cArch3d     c   dqtot(64),dqtot(65),it,weightotal,
cArch3d     c   propsect(1),propsect(2),propsect(3),propsect(4),
cArch3d     c   xlie(50),xlie(51),fin1(50)

cArch3d50   format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
cArch3d     c 1x,i5,1x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
cArch3d     c  e12.6,2X,e12.6,1x,e8.2)
c**************************************************************************

c**
"Arch41_opt_mod_in_initial.dat"*********************************************
****
cArch41       appliedforce = exlk * exfor(29)
cArch41       weightotal=0.d0
cArch41             Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
cd     SEDratio(i)=strainD(i)/weight(i)
cArch41             weightotal=weightotal+weight(i)
cArch4155           Continue

cArch41      if(istep .eq. 1) then
cArch41      write(15,53)
cArch41 53   format('  istep     exlk       appliedforce       a(26)
cArch41    $  a(29)       it     Weight      Area1        Area2
cArch41    $  Area3        Area4      xlie 10       xlie 15       fin1')
cArch41             endif

cArch41      write(15,50) '*',istep,exlk,appliedforce,
cArch41     c   dqtot(26),dqtot(29),it,weightotal,
cArch41     c   propsect(1),propsect(2),propsect(3),propsect(4),
cArch41     c   xlie(10),xlie(15),fin1(20)

cArch4150   format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
cArch41     c 1x,i5,1x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
cArch41     c  e12.6,2X,e12.6,1x,e8.2)
c**************************************************************************

c** "shallow_truss_initial.dat"**********************************************
cshallow       appliedforce = exlk * exfor(38)
cshallow       weightotal=0.d0
cshallow            Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
cd     SEDratio(i)=strainD(i)/weight(i)
cshallow            weightotal=weightotal+weight(i)
cshallow55          Continue

cshallow       if(istep .eq. 1) then
cshallow       write(15,53)
cshallow 53    format('  istep     exlk       appliedforce       a(28)
cshallow     $  a(38)       it     Weight      Area1        Area2
cshallow     $  Area3        Area4      xlie 10       xlie 15       fin1')
cshallow            endif

cshallow      write(15,50) '*',istep,exlk,appliedforce,
cshallow     c   dqtot(26),dqtot(29),it,weightotal,
cshallow     c   propsect(1),propsect(2),propsect(3),propsect(4),
cshallow     c   xlie(16),xlie(20),fin1(20)

cshallow50    format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,
cshallow     c 1x,i5,1x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,
cshallow     c  e12.6,2X,e12.6,1x,e8.2)
c**************************************************************************

c *** frame ***
c** "frame_in.dat" & "frame_out.dat*******************************************

cf                  totald=sqrt(dqtot(1)**2+dqtot(2)**2) !total nodal displ.

cframe              appliedforce = exlk * exfor(1)

cframe       if(istep .eq. 1) then
cframe       write(15,53)
c  53  format('  istep     exlk       appliedforce       dqtot(1)
c    $   dqtot(2)    dqtot(4)  dqtot(5)   it')
cframe   53  format('  istep     exlk       appliedforce       dqtot(1)
cframe     $   dqtot(2)     dqtot(4)     dqtot(5)       it')

cframe       endif


c       write(15,50) '*',dqtot(1)

cframe       write(15,50) '*',istep,exlk,appliedforce,
cframe      c dqtot(1),dqtot(2),dqtot(4),dqtot(5),it
cframe   50  format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,6x,
cframe      c       e10.4,6x,e10.4,3X,i5)

c *** Arch ***
c** "arch42in.dat" & "arch42_3D_in.dat***************************************
c      appliedforce = exlk * exfor(11)
c      if(istep .eq. 1) then
c      write(15,53)
c  53  format('  istep     exlk       appliedforce       dqtot(11)
c    $      it')
c       endif
c       write(15,50) '*',istep,exlk,appliedforce,
c     c dqtot(11),it
c  50   format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,4x,i5)

ccc        write(15,50) exlk
ccc  50    format(e14.6)

ca   53   format('  istep     exlk       appliedforce       dqtot(61)
ca     $   dqtot(62)    dqtot(64)  dqtot(65)   dqtot(85)   dqtot(86)
ca     $      it  ad(61)   ad(62)')
ca       endif
ca       write(15,50) '*',istep,exlk,appliedforce,
ca      c dqtot(61),dqtot(62),dqtot(64),dqtot(65),dqtot(85),dqtot(86),it,
ca      c  ad(62),ad(65)
ca   50   format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
ca      c       3x,e10.4,3x, e10.4,3X e10.4,4x,i5,4x,e10.4,3x,e10.4)


c**************************************************************
c** "t3a_in.dat" 3 member truss ************************
c      appliedforce = exlk * exfor(4)
c     if(istep .eq. 1) then
c     write(15,53)
c 53   format('  istep     exlk       appliedforce       dqtot(4)
c     $   dqtot(5)    dqtot(6)  dqtot(7)   dqtot(8)   dqtot(9)
c     $      it')
c      endif
c     write(15,50) '*',istep,exlk,appliedforce,
c     c dqtot(4),dqtot(5),dqtot(6),dqtot(7),dqtot(8),dqtot(9),it
c50   format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
c     c       3x,e10.4,3x, e10.4,3X e10.4,4x,i5)
c**************************************************************************

c     if(istep .eq. 1) then
c     write(15,53) '*',istep,exlk,appliedforce,
c     c              dqtot(61),dqtot(62),it
c 53   format(a1,2x,i3,5x,f10.6,3x,f10.6,6x,f10.8,6x,f12.8,
c     c       6x,i3)
c                    endif
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              dqtot(61),dqtot(62),it
c 50   format(a1,2x,i3,5x,f10.6,3x,f10.6,6x,f10.8,6x,f12.8,
c     c       6x,i3)
c** "Tripodin.dat  using 3d element"*****************************
c       appliedforce1 = exlk * cxfor(10)
```

```
c       appliedforce2 = exlk * exfor(12)
c     if(istep .eq. 1) then
c     write(15,53)
c 53  format(' istep     exlk      appforcex  appforcez
c    c    a(10)          a(11)         a(12)         it')
c     endif
c     write(15,50) '*',istep,exlk,appliedforce1,appliedforce2,
c    c              dqtot(10),dqtot(11),dqtot(12),it
c50   format(a1,2x,i4,5x,f10.6,3x,e10.3,3x,e10.3,6x,e10.4,6x,e10.4,6x,
c    c      e10.4,4x,i5)
c** "twobar_in.dat"**************************************************
ctwo       appliedforce = exlk * exfor(7)
ctwo      if(istep .eq. 1) then
ctwo      write(15,53)
ctwo 53   format(' istep     exlk      appliedforce      a(5)
ctwo    $    a(7)     it        ad(5)      ad(7)')
ctwo      endif
ctwo      write(15,50) '*',istep,exlk,appliedforce,
ctwo     c    dqtot(5),dqtot(7),it,
ctwo     c    ad(5),ad(7)
ctwo50    format(a1,2x,i4,5x,e14.6,3x,e10.4,6x,e10.4,4x,e10.4
ctwo     c          ,4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
c** "snap4_in.dat"*****************************************************
csnap4       appliedforce = exlk * exfor(11)
csnap4      if(istep .eq. 1) then
csnap4      write(15,53)
csnap4 53   format(' istep     exlk      appliedforce      a(1)
csnap4    $    a(2)       a(11)     it        ad(3)      ad(4)')
csnap4      endif
csnap4      write(15,50) '*',istep,exlk,appliedforce,
csnap4     c    dqtot(1),dqtot(2),dqtot(11),it,
csnap4     c    ad(1),ad(2)
csnap450    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
csnap4     c          4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
c** "snap_in.dat"******************************************************
csnap       appliedforce = exlk * exfor(11)
csnap      if(istep .eq. 1) then
csnap      write(15,53)
csnap 53   format(' istep     exlk      appliedforce      a(1)
csnap    $    a(2)       a(11)     it        ad(3)      ad(4)')
csnap      endif
csnap      write(15,50) '*',istep,exlk,appliedforce,
csnap     c    dqtot(1),dqtot(2),dqtot(11),it,
csnap     c    ad(1),ad(2)
csnap50    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
csnap     c          4x,i5,4x,e10.4,3x,e10.4)
c*********************************************************************
c** "domein.dat"*******************************************************
cd       appliedforce = exlk * exfor(3)
cd       weightotal=0.d0
cd                  Do 55 i=1,nel
c                   print *,'istep=',istep
c                   print *,'i=',i
c                   print *,'weightotal=',weightotal
c                   print *,'weight(_)=',weight(i)
c                   pause
cd      SEDratio(i)=strainD(i)/weight(i)
cd                  weightotal=weightotal+weight(i)
cd55                Continue

cd      if(istep .eq. 1) then
c       write(15,53)
c 53   format(' istep     exlk      appliedforce      a(1)
c    $    a(2)       a(3)       a(4)       a(5)       a(6)
c    $       it          ad(3)      ad(4)')
cd      write(15,53)
cd 53   format(' istep     exlk      appliedforce      a(1)
cd    $  a(2)       a(3)         it        Weight     Area1    Area2
cd    $       Area3     Area4')

cd      write(22,60)
cd60                format('SDratio(1)  SDratio(2)  SDratio(3)  SDratio(4)
cd     $ SDratio(5)  SDratio(6)  SDratio(7)  SDratio(8)  SDratio(9)
cd     $ SDratio(10)  SDratio(11)  SDratio(12)  SDratio(13)
cd     S SDratio(14)  SDratio(15)')


cd      write(22,58) SEDratio(1),SEDratio(2),SEDratio(3),SEDratio(4),
cd     $ SEDratio(5),SEDratio(6),SEDratio(7),SEDratio(8),SEDratio(9),
cd     $ SEDratio(10),SEDratio(11),SEDratio(12),SEDratio(13),
cd     S SEDratio(14),SEDratio(15)


c     write(18,59)
c59                 format('strainD(1)  strainD(2)  strainD(3)  strainD(4)
c     $ strainD(5)  strainD(6)  strainD(7)  strainD(8)  strainD(9)
c     $ strainD(10)  strainD(11)  strainD(12)  strainD(13)  strainD(14)
c     $  strainD(15)')

c                   write(18,58) strainD(1),strainD(2),,strainD(3),strainD(4),
c     $strainD(5),strainD(6),strainD(7),strainD(8),strainD(9),strainD(10)
c     $,strainD(11),strainD(12),strainD(13),strainD(14),strainD(15)

cd 58   format(e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,
cd     $ e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,
cd     $ 2x,e10.2,2x,e10.2)

cd      write(21,54)
cd 54   format(' istep      exlk      ad(1)      ad(2)      ad(3)
cd     $      ad(4)      ad(5)      ad(6)      ad(7)      ad(8)
cd     $      ad(9)      ad(10)     ad(11)     ad(12)     ad(13)
cd     $      ad(14)     ad(15) resk(1) resk(2) resk(3)')

cd                  write(21,56) istep,exlk,ad(1),ad(2),ad(3),ad(4),ad(5),ad(6),
cd     $ ad(7),ad(8),ad(9),ad(10),ad(11),ad(12),ad(13),ad(14),
cd     $ ad(15),resk(1),resk(2),resk(3)
cd 56   format(i4,2x,e14.6,2x,e10.2,2x,e10.2,2x,e10.2,2x,
cd     $e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,
cd     $e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,
cd     $e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2,2x,e10.2)

cd                  write(25,68) istep,exlk,Anew(1),Anew(7),Anew(13),Anew(14)
cd 68   format(i4,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6)

cd      endif

cd      write(15,50) '*',istep,exlk,appliedforce,
cd     c   dqtot(1),dqtot(2),dqtot(3),it,weightotal,
cd     c   propsect(1),propsect(2),propsect(3),propsect(4)

cd50    format(a1,2x,i4,2x,e14.6,4x,e10.2,4x,e10.4,4x,e10.4,4X,e10.2,
cd     c         2x,i5,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4)

c                   print *,'resk(3) inside result=',resk(3)
c                   pausc

cd      write(22,58) SEDratio(1),SEDratio(2),SEDratio(3),SEDratio(4),
cd     $ SEDratio(5),SEDratio(6),SEDratio(7),SEDratio(8),SEDratio(9),
cd     $ SEDratio(10),SEDratio(11),SEDratio(12),SEDratio(13),
cd     S SEDratio(14),SEDratio(15)


c                   write(19,57) resk(1),resk(2),resk(3),resk(4),
c     $resk(5),resk(6),resk(7),resk(8),resk(9),resk(10)
c     S,resk(11),resk(12),resk(13),resk(14),resk(15)

cd                  write(19,57) Fin1(1),Fin1(2),Fin1(3),Fin1(4),
cd     $Fin1(5),Fin1(6),Fin1(7),Fin1(8),Fin1(9),Fin1(10)
cd     $,Fin1(11),Fin1(12),Fin1(13),Fin1(14),Fin1(15)

cd                  write(17,57) strainE(1),strainE(2),strainE(3),strainE(4),
cd     $strainE(5),strainE(6),strainE(7),strainE(8),strainE(9),strainE(10)
cd     $,strainE(11),strainE(12),strainE(13),strainE(14),strainE(15)

c                   write(16,57) strainD(1),strainD(2),strainD(3),strainD(4),
c     $strainD(5),strainD(6),strainD(7),strainD(8),strainD(9),strainD(10)
c     $,strainD(11),strainD(12),strainD(13),strainD(14),strainD(15)

cd 57   format(e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,
cd     $ e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,
cd     $ 2x,e10.4,2x,e10.4)

cc                  If(istep .eq. 7920)then
c                   If(istep .eq. 7420)then


cd                  write(18,67) exlk,strainD(1),strainD(2),strainD(3),strainD(4),
cd     $strainD(5),strainD(6),strainD(7),strainD(8),strainD(9),strainD(10)
cd     $,strainD(11),strainD(12),strainD(13),strainD(14),strainD(15)
cd 67   format(e10.4,2xe10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,
cd     $ e10.4,2x,
cd     $ e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,2x,e10.4,
cd     $ 2x,e10.4,2x,e10.4)

c                   write(25,68) istep,exlk,Anew(1),Anew(7),Anew(13),Anew(14)
c 68   format(i4,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6,2x,e12.6)

cc                  Endif

cdome       write(15,50) '*',istep,exlk,appliedforce,
cdome     c    dqtot(1),dqtot(2),dqtot(3),dqtot(4),dqtot(5),dqtot(6),it,
cdome     c    ad(3),ad(4)
cdome 50    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
cdome     c         3x,e10.4,3x, e10.4,3X e10.4,4x,i5,4x,e10.4,3x,e10.4)
c*************************************************************
c** "truss7_3d_in.dat  using 3d element"**************************
C** "truss11_3d_in.dat using 3d element"**************************
ctruss       appliedforce = exlk * exfor(8)
ctruss      if(istep .eq. 1) then
ctruss      write(15,53)
ctruss 53   format(' istep     exlk      appliedforce      a(1)
ctruss    $  a(2)       a(4)       a(5)       a(7)       a(8)
ctruss    $  it')
ctruss      endif
ctruss      write(15,50) '*',istep,exlk,appliedforce,
ctruss     c         dqtot(1),dqtot(2),dqtot(4),dqtot(5),dqtot(7),dqtot(8),it
ctruss50    format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,3X,f10.4,
ctruss     c         f10.4,3x, f10.4,3X f10.4,4x,i5)

c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(6),a(9),a(3),it
c 50   format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,
c     c         6x,i3)
cc 50   format(a1,2x,i3,5x,f10.6,3x,f10.2,6x,f10.4,6x,f10.4,f10.4,
cc     c         6x,i3)
c     print *,'check 3 inside result'
c     write(*,50) '*',istep,exlk,appliedforce,
c     c              a(3),a(4),a(5),a(6),a(13),a(14),it
c Below IS FOR T3y
c     write(15,50) '*',istep,exlk,appliedforce,
c     c              a(1),a(2),a(3),a(4),it,a(12)
c     write(15,50) '*',istep,exlk,xinternalforce,yinternalforce,
c     c              a(3),a(4),it

c     stop
ccc     write(12,51) xinternalforce,yinternalforce
ccc 51   format(f12.2,1x,f12.2)
c     write(12,51) appliedforce
c 51   format(f12.2)

c     write(13,52) a(3)
cc     write(13,52) a(5)
c18     write(13,52) a(20)
carch     write(13,52) a(44)
cc 52   format(f10.6)


c 51   continue
c
cc 10   format(t5,'Load',t13,'Load',
cc     $       t28,'External',t62,'No. of')
cc 20   format(t5,'Step',t13,'Multiplier',
```

```
cc    $       t28,'Load',t45,'Displacement',
cc    $       t62,'Iter.')
cc 30   format(t5,i3,t13,f10.6,t28,f10.6,t45,f12.6,t62,i3)
c
c       print *,'end of result'
c       pause
c       stop

c     Endif

      return
      end

C*********************************************************************
*
C***                SUBROUTINE: ITNRML                          ***
C*********************************************************************
*
c     subroutine itnrml(itemax,toldis,tolfor,dli,dlim1,
c     $                 exlk,iconv,iflag,it,nc,iplvl)
      subroutine itnrml(itemax,dli,dlim1,b,exfor,ad,neltypc,qk,
     $               iflag,it,ne,iplvl,ndof,ia,ncl,ncoef1,jeall,
c     $                iflag,it,nc,iplvl,ndof,ia1,ia2,nel,ncoef1,jcall,
     $               ieall,ie,je,iet,jet,maxdofpe,dqtot,dqi,dqim1,
     $               ja,an,propmat,propsect,x,y,z,ndofpn,numnodes,
     $               ip,iu,disp,ianal,ndofpe,iskip,itempo2,iarc,lm,
c     $                itnrmlon,iboundc,esm,elk)
     $               itnrmlon,iboundc,ae,di,istep,iarea,
     $               strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $               Fin2,VED,VEDold,xlie,exlkdl0,exlkdli,exlkdlk,iexlk)
c                       use

      implicit real*8(a-h,o-z)
c     common/RIK/ stifi(20,20),dqi(500),dqim1(50),dqk(50),dqk1(50),
c     $         dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
c     $         resk(50),scali(50),work1(50),indx(50),neq
C     common/RIK/ stifi(20,20),dqim1(50),dqk(50),dqk1(50),
C     $         dqk2(50),dqtot(50),exfor(50),qk(50),fintk(50),
C     $         resk(50),scali(50),work1(50),indx(50),neq
      common/RIK/ stifi(60,60),dqk1(5000),
     $    dqk2(5000),fintk(5000),
     $    resk(5000),scali(5000),work1(5000),indx(5000),neq


c       common/deltaqi/dqi(500)

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
      common/disp/ux(2),uy(2),uz(2)
      common/infor/getforce,finx(2),finy(2),finz(2),getstiff,getsolve
      common/zvector/za(1500),za1(1500),za2(1500),za3(1500),ipick
      common/testing/toldis,tolfor,exlk,iconv

      dimension ie(*),je(*),iet(*),jet(*),b(*),exfor(*),nel(*)
c     dimension dqtot(*),dqi(*),dqim1(*),ad(*),qk(*)
c     dimension dqtot(*),dqim1(*),qk(*),ad(*)
      dimension dqtot(*),dqim1(*),ad(*),qk(*)
                      dimension ja(*),an(*),disp(*)
c                     dimension an(*),disp(*)
                      dimension propmat(1),propsect(1),x(*),y(*),ip(*),iu(*)
      dimension itempo2(*),lm(1),ae(1)
                      dimension iboundc(*),iarea(1)

                      dimension dqi(5000),dqk(5000)
                      dimension strainel(*),strainE(*),strainD(*),weight(*),Anew(*),
     $         Apart(*),Fin1(*),Fin2(*),VED(*),VEDold(*),xlie(*)

c     dimension itempo2(*),lm(1), elk(maxdofpe,maxdofpe)

c     dimension stifi(neq,neq),dqi(neq),dqim1(neq),dqk(neq),
c     $         dqk1(neq),dqk2(neq),exfor(neq),fintk(neq),qk(neq),
c     $         resk(neq),scali(neq),work1(neq)
c
c                     deallocate (ja)

      itnrmlon=1 !Let everyone know we are going thru "itnrml"
c               "syseqn" "& "assembly" need to do something more

c     print *,'!!!!!!!!!!!!!!!!!  starting itnrml  !!!!!!!!!!!!!!!!!'
c                     pause
c                     print *,'itempo2(1) inside itrnml=',itempo2(1)
c                     pause

c                     print *,'ad stiff=',(ad(i),i=1,maxdofpe**2)
c                     pause

c     print *,'ipick = ',ipick
c                     print *,'istep=',istep
c                     print *,'it=',it
c     print *,'getforce =',getforce
c     print *,'getstiff =',getstiff
c                      print *,'qk(-) start of itnrml=',(qk(i),i=1,ndof)
c                      print *,'b start of itnrml=',(b(i),i=1,ndof)
c                      print *,'ad(_) start itnrml=',(ad(ic),ic=1,ndof)
c                     print *,'an(_) start itnrml=',(an(ic),ic=1,50)
c                     pause
c       stop
c     print *,'a at start of itnrml = ',a
c     print *,'exlk at start of itnrml = ',exlk
c     print *,'dqi at start of itnrml = ',(dqi(i),i=1,ndof)
c       print *,'exfor at start of itnrml =',(exfor(i),i=1,ndof)
c                     pause
c       stop
      dlk = 0.d0
c     dli             =0.d0
      iconv = 1
      iflag = 0
c     it = 0
c
      do 20 i=1,neq
c       do 10 j=1,neq
c         stifi(j,i) = 0.d0
c 10    continue
c       print *,'exfor(i) start =',exfor(i)
        dqk(i) = 0.d0
        dqk1(i) = 0.d0
        dqk2(i) = 0.d0
        fintk(i) = 0.d0
        work1(i) = 0.d0
c       print *,'exfor(i) end =',exfor(i)
 20   continue
c** set counter equal to zero for next "it"
      iam = 0
c     print *,'exfor =',exfor
c     stop
c     print *,'iconv = ',iconv
c     print *,'itemax = ',itemax
c     print *,'it = ',it
c                     pause
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1002 if (iconv .ne. 0 .and. it .le. itemax) then
c       print *,'iconv =',iconv
c       Print *,'it = ',it
c       print *,'itemax =',itemax
c                     stop
        it = it + 1
c       write(*,*) 'Beginning Iteration Number:',it
c                     stop
c ... compute Ktk for eqn. 4.97
cc      print *,'ne & iplvl inside itnrml = ',ne,iplvl
c     print *,'qk inside itnrml=',qk
c c     pause
c       stop
c       Getforce=1
c**Only get stiffness
                      Getsolve=0
                      Getforce=0
        Getstiff=1
c       print *,'exfor = ',exfor
c                     print *,'chk before syseqn inside itnrml'
c                     write(6,*)'chk before syseqn inside itnrml'
c       stop

c                     deallocate (ja)
c                     deallocate (itempo1)
c                     deallocate (an)
c                     print *,'********** iskip in itnrml **************'
c                     print *,'********** iskip in itnrml **************'
c                     pause
c                     print *,'iskip in itnrml before 1st syseqn=',iskip
c                     pause
c                     print *,'iskip in itnrml before 1st syseqn=',iskip
c                     pause
c                     print *,'iskip in itnrml before 1st syseqn=',iskip
c                     pause
c                     print *,'iskip in itnrml before 1st syseqn=',iskip
c                     pause
c                     print *,'getforce in itnrml before 1st syseqn=',getforce
c                     print *,'getstiff in itnrml before 1st syseqn=',getstiff
c     pause
c                     print *,'********************'
c                     print *,'********************'
c                     print *,'disp before 1st syseqn in itnrml=',(disp(i),i=1,ndof)
c                     print *,'********************'
c                     print *,'********************'
c                     pause
c                     print *,'ad stiff before=',(ad(i),i=1,ndof)
c                     pause
c                     print *,'an stifl=',(an(i),i=1,maxdofpe**2)
c                     print *,'ae(_)before 1st syseqn in itnrml=',(ae(ic),ic=1,18)
c                     print *,'b(_) before=',(b(ic),ic=1,ndof)
c                     pause
c     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                     pause
c                     pause

                      call syseqn(nc,iplvl,b,ndof,disp,iarc,qk,
     $       icall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $       neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $       iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqim1,
     $       ux,uy,uz,di,it,area,youngm,iarea,
     $       ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $       jallo,
     $       strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $       Fin2,VED,VEDold,xlie,SED)

c     $ja,iboundc,ndofpcd,bc,lm,ndofpcd,ad,ip,tempo1,dm,am,ianal,lumpmass)
c       print *,'dqk1 after syseqn in intml=',dqk1
c                     print *,'********************'
c                     print *,'********************'
c                     print *,'disp after 1st syseqn in itnrml=',(disp(i),i=1,ndof)
c                     pause
c                     print *,'b after 1st syseqn in itnrml=',(b(i),i=1,ndof)
c                     pause
c                     print *,'ad stiff after 1st syseqn=',(ad(i),i=1,ndof)
c                     pause
c                     print *,'an(_) after 1st syseqn in itnrml=',(an(ic),ic=1,50)
c                     print *,'ae stiff after 1st syseqn=',(ae(i),i=1,maxdofpe**2)
c                     print *,'********************'
c                     print *,'********************'
c                     pause

c     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                     pause

c       If(it .eq. 2)then
c     print *,'it = ',it
c                     print *,'disp after 1st syseqn in itnrml=',(disp(ic),ic=1,ndof)
c                     print *,'b(_) after 1st syseqn in itnrml=',(b(ic),ic=1,ndof)
c                     print *,'ad stiff after 1st syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                     print *,'an(_) after 1st syseqn in itnrml=',(an(ic),ic=1,50)
c       stop
c       endif
```

```
c                     stop
c     pause
c                       print *,'chk after syseqn inside itnrml'
c                       write(6,*)'chk after syseqn inside itnrml'
c                     write(6,*)'b after 1st syseqn in itnml=',(b(i),i=1,neq)
c                     write(6,*)'disp after 1st syseqn in itnml=',(disp(i),i=1,neq)

c       stop

c     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                     pause

c^^^^^^ SETTING ZERO DIAGONAL VALUES TO ONE ^^^^^^^^^^^^^^^^^
c       Do 1014 i=neq*2+1,neq*3
c       Do 1014 i=1,neq
c       If(ad(i) .GT. .00001) GoTo 1014
c       If(ad(i) .GT. 1.e-8) GoTo 1014
c         ad(i)=1.e-8
c       ad(i)=0.d0
c1014   Continue

c                     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                     pause

c???          do 702 i=1,neq
c               scali(i) = 1.d0/stifi(i,i)
c     c c     print *,'a(neq*2+i) = ',a(neq*2+i)
cc                scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
c???              scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c???                  print *,'scali =',scali(i)
c???                  write(6,*)'scali =',scali(i)
c??? 702        continue

c**I call modify to get the stiffness w/delected rows/columns
cc        call modify
c       print *,'dqk1 after modify in intml=',dqk1
c       stop
c     print *,'Check after fifth modify'
c     print *,' "a" after 5th modify = ',a
c     c     pause
c c     print *,'ibstor = ',ibstor
c c     print *,'bvstor = ',bvstor
c       stop

        Getstiff = 0 ! set getting just stiff back to zero

cc      print *,'a after syseqn inside itnrml = ',a
c         If(it .eq. 2)then
c     print *,'it = ',it
c     print *,'a at it=2 ',a
c         stop
c         endif
c       call stiff

cc      print *,'stifi inside itnrml = ',stifi
c ... compute (dqk)1 using eqn. 4.97
c ... compute (dqk)2 using eqn. 4.99
c     print *,'exfor(_) = ',(exfor(ic),ic=1,ndof)
c                     pause
c       stop

        do 30 i=1,neq
          dqk1(i) = exfor(i)
c           dqk1(i) = b(i)
          dqk2(i) = resk(i)
c       print *,'exfor(i) = ',exfor(i)
c     print *,'dqk1(i) & i = ',dqk1(i),i
c     print *,'resk(i) & i = ',resk(i),i
c     print *,'dqk2(i) & i = ',dqk2(i),i
 30     continue
c                     pause
c     print *,'dqk1(_)=',(dqk1(ic),ic=1,neq)
c                     pause
c     print *,'dqk2(_)=',(dqk2(ic),ic=1,neq)
c     pause
c       stop

c c     print *,'resk = ',resk

c put dqk1(i) into big "a(i) as a force, get out as displacemnet"
        j=0
cc      print *,'neq, neq2, np2 = ',neq,neq2,np2
c       stop
c       print *,'a =',a
c       stop
c**********************************************
c       If(ityp .eq. 6)Then
cc        do 31 i=neq+1,neq*2

        do 31 i=1,neq
cc          j=j+1
c           a(i)=dqk1(j)

          b(i)=dqk1(i)
c                       print *,'b(i) before new 2nd syseqn=',b(i)
c                       pause
c c     print *,'a(i) i = ',a(i),i
c     print *,'dqk1(i) = ',dqk1(i)
 31     continue
c     print *,'b(_) after dqk1(_) = ',(b(ic),ic=1,neq)
c                     pause

CXX                   write(6,*)'b(i) inside itnml=',(b(i),i=1,neq)
c       Endif
c       If(ityp .eq. 12)Then
c       do 34 i=neq+1,neq*3
c         j=j+1
c         a(i)=dqk1(j)
c c     print *,'a(i) i = ',a(i),i
c c     print *,'dqk1(j) j = ',dqk1(j),j
c 34     continue

c       Endif
c**********************************************

c       print *,'dqk1 = ',dqk1
c       stop
c        call solve(dqk1)
cc  c     print *,'n before ludcmp = ',n
c          stop
cc  c     print *,'****ludcmp 3****'
c       c     pausc

cc      write(*,*) 'Before dcmpbd inside itnrml'
c     print *,'a = ',a
c     stop
c** solve for new "dqk1"

      ipick = 1

cc      call dcmpbd
cc      print *,'after dcmpbd'
c       print *,'a after dcmpbd inside itnrml =',a
c       stop
cc      write(*,*) 'After dcmpbd inside itnrml'
cc      call slvbd
c     write(*,*) 'After slvbd1 & ipick=1 inside itnrml'
c       stop
c                     print *,'############ NEW SYSEQN ADDED itnrml############'
c                     print *,'############ 2nd Syseqn in itnrml ############'
c                     pause

c                     print *,'********************'
c                     print *,'********************'
c                     print *,'disp before 2nd syseqn in itnrml=',(disp(i),i=1,ndof)
c                     pause
c                     print *,'b before 2nd syseqn in itnrml=',(b(i),i=1,ndof)
c                     pause
c                     print *,'ad stiff before 2nd syseqn=',(ad(i),i=1,maxdofpe**2)
c                     pause
c                     print *,'an(_) before 2nd syseqn in itnrml=',(an(ic),ic=1,50)
c                     pause
c                     print *,'ae stiff after 1st syseqn=',(ae(i),i=1,maxdofpe**2)
c                     print *,'********************'
c                     print *,'********************'
c
c... I only want to solve, I don't want update forces or stiff
                        getforce=0
                        getstiff=0
                        getsolve=1
c                     print *,'getsolve=1 inside itnrml before 2nd syseqn'
c                     pause
c...Need to solve eqns only, no new loads or stiff!
        call syseqn(ne,iplvl,b,ndof,disp,iarc,qk,
     $              ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $              neltype,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $              iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqiter1,
     $              ux,uy,uz,di,it,area,youngm,iarea,
     $           ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $           jallo,
     $           strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $           Fin2,VED,VEDold,xlie,SED)

c                     print *,'############## END SYSEQN ##############'
c                     pause
c                     print *,'############ After 2nd Syseqn in itnrml ##############'
c                     pause
c                     pause
c                     pause
c                     print *,'neq=',neq
c                     print *,'disp(_) after 2nd syseqn solve =',(disp(ic),ic=1,ndof)
c                     pause
c                         print *,'b(i) after new 2nd syseqn=',(b(ic),ic=1,ndof)
c                         pause
c                     print *,'ad after 2nd syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                                 print *,'an after 2nd syseqn solvs=',(an(i),i=1,50)
c                     pause

c         If(it .eq. 2)then
c     print *,'it = ',it
c                     print *,'disp after 2nd syseqn in itnrml=',(disp(ic),ic=1,ndof)
c                     print *,'b(_) after 2nd syseqn in itnrml=',(b(ic),ic=1,ndof)
c                     print *,'ad stiff after 2nd syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                     print *,'an(_) after 2nd syseqn in itnrml=',(an(ic),ic=1,50)
c         stop
c         endif

c                     stop
c                     print *,'ae after 2nd syseqn=',(ae(i),i=1,maxdofpe**2)
c                     pause
c                     pause
c                     print *,'dqk1(_)=',(dqk1(ic),ic=1,9)
c                     pause
c** set "dqk1" equal to the solution displ stored in "a"
      DO 178 LL=1,neq
cc      dqk1(LL) = a(LL)
      dqk1(LL) = disp(LL)
c                     print *,'dqk1(_) after new syseqn=',dqk1(LL)
 178   continue
c                     pause
c                     write(6,*)'dqk1(LL) inside itnml=',(dqk1(LL),i=1,neq)
c     print *,'a = ',a
c     print *,'dqk1 = ',dqk1
c     stop
c** new dqk1 from big "a"
cc      print *,'neq = ',neq
c       stop
c** get out "dqk1" as displacement, now put back into "a" at displace
c** NO leave dqk1 in own vector, donot put into "a"
c         do 177 ink=1,neq
c           dqk1(ink)=a(ink)
c c     print *,'a(ink) = ',a(ink)
c c     print *,'dqk1(ink) & ink = ',dqk1(ink),ink
c 177     continue
```

```
c      stop

c           call ludcmp(stifi,n,np,indx,X)
cc  c      print *,'h after ludcmp = ',n
c          stop
c           call lubksb(stifi,n,np,indx,dqk1)
c                    print *,'np=',np
c                    print *,'neq=',neq
c                    pause
c            do 12 i=1,np
           do 12 i=1,neq
 12          indx(i)=0

c** Now do the same for dqk2

c       call solve(dqk2)

c put dqk2(i) into big "a(i) as a force, get out as displacemnet"
       j=0
cc     print *,'neq, neq2, np2 = ',neq,neq2,np2
c       stop
c*******************************************************
c       If(ityp .eq. 6)then
        do 32 i=1,neq
c         j=j+1
c         a(i)=dqk2(j)
                        b(i)=dqk2(i)
c c     print *,'a(i) = ',a(i)
c      print *,'dqk2(i) = ',dqk2(i)
 32     continue
c                    print *,'#### After 3rd Syseqn inside itnrml ####'
c                    pause

c                    If(istep.gt.900)Then
c                    print *,'dqk2(_) inside itnrml=',(dqk2(ic),ic=1,neq)
c                    print *,'b(_) inside itnrml=',(b(ic),ic=1,neq)
c                    pause
c                    Endif

c                    write(6,*)'dqk2(_) inside itnrml=',(dqk2(ic),ic=1,neq)
c       Endif

c       If(ityp .eq. 12)then
c        do 33 i=neq+1,neq*3
c          j=j+1
c          a(i)=dqk2(j)
c c      print *,'a(i) = ',a(i)
c c      print *,'dqk2(j) = ',dqk2(j)
c 33      continue
c       Endif
c*******************************************************

c        stop

cc      write(*,*) 'Before dcmpbd inside intml 2nd time'
c      stop
       ipick = 2

**********************


* Not sure if "a" is getting rewritten and used properly?????


**********************
cc      print *,'a before dcmpbd =',a
c       print *,'dqi before dcmpbd = ',dqi
c       print *,'dqk1(_) inside intml =',(dqk1(ic),ic=1,neq)
c                    pause
c       stop
cc      call dcmpbd
c      stop
c      write(*,*) 'After dcmpbd inside intml 2nd time'
cc      call slvbd
c      write(*,*) 'After slvbd & ipick=2 inside intml 2nd time'
c                    print *,'################ END SYSEQN 3 ################'
c... I only want to solve, I don't want update forces or stiff
                       getforce=0
                       getstiff=0
                       getsolve=1
c                    print *,'getsolve=1 inside itnrml before 3rd syseqn'
c                    pause

c                    print *,'b(i) before new 3rd syseqn=',(b(ic),ic=1,ndof)
c                    pause
c                    print *,'ad before 3rd syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                    pause
c                    print *,'disp(_) before 3rd syseqn solve only=',(disp(ic),ic=1,ndof)
c                    pause
c...Need to solve eqns only, no new loads or stiff!
        call syseqn(nc,iplvl,b,ndof,disp,iarc,qk,
     $              icall,ie,je,iet,jet,jeall,ia,ac,maxdofpe,
     $              neltypc,nel,ndofpe,x,y,z,propmat,propsect,lm,
     $              iboundc,ad,an,ipermm,iu,ip,iut,dqtot,dqi,dqim1,
     $              ux,uy,uz,di,it,area,youngm,iarea,
     $          ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $          jallo,
     $          strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $          Fin2,VED,VEDold,xlie,SED)

c                    print *,'################ END SYSEQN 3 ################'
c                    pause
c                    print *,'neq=',neq

c                    print *,'b(i) after new 3rd syseqn=',(b(ic),ic=1,ndof)
c                    pause
c                    print *,'ad after 3rd syseqn solve=',(ad(i),i=1,maxdofpe**2)
c                    pause
c                    print *,'an after 2nd syseqn solvs=',(an(i),i=1,50)
c                    stop

c        If(it .eq. 2)then
c     print *,'it = ',it
c                    print *,'disp after 3rd syseqn in itnrml=',(disp(ic),ic=1,ndof)
c                    print *,'b(_) after 3rd syseqn in itnrml=',(b(ic),ic=1,ndof)
c                    print *,'ad stiff after 3rd syseqn=',(ad(ic),ic=1,maxdofpe**2)
c                    print *,'an(_) after 3rd syseqn in itnrml=',(an(ic),ic=1,50)
c         stop
c        endif

c                    print *,'disp(_) after 3rd syseqn solve only=',(disp(ic),ic=1,ndof)
c                    pause

c** set "dqk1" equal to the solution displ stored in "a"
      DO 179 LL=1,neq
cc       dqk2(LL) = a(LL)
      dqk2(LL) = disp(LL)
c     print *,'dqk2 = ',dqk2(LL)
 179  continue
cc      print *,'a after slvbd = ',a
c     print *,'dqk1 = ',dqk1
c     print *,'dqk2 = ',dqk2
c                    pause
c       stop
c          call stiff
ccccc           call syseqn(ne,iplvl)
c       stop
           do 13 i=1,np
 13          indx(i)=0

c ... compute dlk using eqn. 4.101
       dprd1 = 0.d0
       dprd2 = 0.d0
       do 40 i=1,neq
c                    print *,'dqi(_) dqk1(_) dqk2(_) = ',dqi(i),dqk1(i),dqk2(i)
        dprd1 = dprd1 + dqi(i)*dqk2(i)
        dprd2 = dprd2 + dqi(i)*dqk1(i)
                    If(istep .eq. 1)Then
                      dprd1 = 0.d0
                    endif

 40     continue
c                    print *,'dprd1 inside itnrml=',dprd1
c                    print *,'dprd2 inside itnrml=',dprd2

c                    If(istep .eq. 1)Then
c     print *,'dqi(i)=',dqi
c                    print *,'istep=',istep
c     print *,'dqi(i)=',(dqi(ic),ic=1,ndof)
c     print *,'dqk1(i)=',(dqk1(ic),ic=1,ndof)
c     print *,'dqk2(i)=',(dqk2(ic),ic=1,ndof)
c                    print *,'dprd1=',dprd1
c                    print *,'dprd2=',dprd2
c                    print *,'ad(_)=',(ad(ic),ic=1,ndof)
c                    pause
c                    Endif

c                    Testv2=dlk*(dprd2+dlim1)
c                    print *,'istep=',istep
c                    print *,'Testv2=',testv2
c                    pause

c                    If(Testv2.lt.-1.6e-5)Then
c                    dli=-1.*dli
c                    Else
c                    dli=1.*dli
c                    Endif

c                    If(dprd1 .lt. 0.0 .and. dprd2 .lt. 0.0)Then
c                     If(dprd1 .gt. -.00001 .and. dprd2 .gt. -.1)Then
c                     dli=-1.*dli
c                     Endif
c                    Else
c                     dli=1.*dli
c     Endif

c     print *,'dprd1 = ',dprd1
c     print *,'dprd2 inside itnrml= ',dprd2
c                    pause

cc      print *,'dli inside itnrml= ',dli
c      stop
c      stop

       dlk = (dprd1)/(dprd2 + dli)

c     print *,'dlk inside itnrml = ',dlk
cc    stop
c ... update the external load multiplier exlk
c     print *,'exlk before = ',exlk

c                                        If(istep .gt. 561)then
c                                        dlk = -1.d0*dlk
c                                        Endif
                    If(exlk .GT. .98)then
c                   print *,'after exlk 3 chk'
c                   print *,'resk(_)=',(resk(ic),ic=1,neq)
c                   pause
                    endif

                     IF(exlk .LT. exlk-dlk)then
                    exlkdlk=exlk-dlk

                    iexlk=3

c                    print *,'After 3rd exlk check'
c                   print *,'exlk=',exlk
c                   print *,'exlk-dlk=',exlkdlk
c                   print *,'ad(i) after exlk=',(ad(i),i=1,ndof)
c                   print *,'resk(_)=',(resk(ic),ic=1,neq)
c                   pause
c                   deallocate (dqtot)
c                   deallocate (dqi)
c                   deallocate (dqim1)
```

```
c     pause
                    GOTO 7000

c                   STOP
                    ENDIF

                    If(istep .eq. 1)then
c                   print *,'istep = 1'
c                   print *,'exlk=',exlk
c                   print *,'dlk=',dlk
c                   pause
                    Endif

      exlk = exlk - dlk

c                   print *,'exlk=',exlk
c                   pause

c                                   If(istep .eq. 561)then
c                                   exlk = -1.d0*exlk
c                                   Endif


c     print *,'1st exlk in itnrml = ',exlk
c                   pause
c                   print *,'dlk in itnrml=',dlk
c                   pause
c     write(6,*)'1st exlk in itnrml = ',exlk
c                   write(6,*)'dlk in itnrml=',dlk
c                   stop

ccc     pause

c       stop
c ... update dlim1 for next load step
        dlim1 = dlim1 - dlk
c       stop
c ... compute dqk using eqn. 4.94
c ... update dqim1 for next load step
c ... update the total displacements using eqn. 4.103
c     print *,'neq = ',neq
c     stop
      do 50 i=1,neq
c                   If(istep.eq.914)Then
c       dqk(i) = dqk2(i) + dlk*dqk1(i)
c      Else
       dqk(i) = dqk2(i) - dlk*dqk1(i)
c                   Endif
c                   If(dqk(11).lt.0)Then
c                   print *,'istep=',istep
c     print *,'dqk(11)=',dqk(11)
c                   pause
c                   Endif
c       stop
c       dlim1(i) = dlim1(i) + dqk(i)
        dqim1(i) = dqim1(i) + dqk(i)
c     print *,'dqim1(i) & i = ',dqim1(i),i
c                   print *,'qk(11)before=',qk(11)
        qk(i) = qk(i) + dqk(i) !Update Total Displacements
c     print *,'qk(11) after=',qk(11)
 50     continue

c                   print *,'istep=',istep
c     print *,'dqk(11)=',dqk(11)
c     print *,'qk(11)=',qk(11)
c                   pause

c                   If(istep.gt.900)Then
c                   print *,'istep=',istep
c                   print *,'dqi(_)=',(dqi(ic),ic=1,neq)
c                   print *,'dqk(_)=',(dqk(ic),ic=1,neq)
c                   print *,'qk(_)=',(qk(ic),ic=1,neq)
c                   print *,'dli(_)=',(dli(ic),ic=1,neq)
c                   print *,'dlk(_)=',(dlk(ic),ic=1,neq)
c                   print *,'ad(_)=',(ad(ic),ic=1,neq)
c                    pause
c     endif
c       Do 5050 ii=1,neq
c         dqim1(ii) = dqim1(ii) + dqk(ii)
c5050               continue
cc      If(istep.gt.911)Then
c     If(istep.gt.136)Then
cc                  print *,'istep=',istep
cc                  print *,'dqk1(i)=',(dqk1(ic),ic=1,neq)
cc                  print *,'dlk=',dlk
cc                  print *,'dqk2(i)=',(dqk2(ic),ic=1,neq)
cc      print *,'dqk(i)=',(dqk(ic),ic=1,neq)
cc                  print *,'qk(i)=',(qk(ic),ic=1,neq)
cc                  print *,'ad(ic)=',(ad(ic),ic=1,neq)
cc                  pausc
cc                  Endif
c                   write(6,*)'qk(i)=',(qk(i),i=1,neq)
c                   print *,'qk(i)=',(qk(ic),ic=1,neq)
c                   pause
c       stop

c     print *,'exlk =',exlk
cc     print *,'dlim1 =',dlim1
cc     print *,'dqk=',dqk
cc     print *,'qk=',qk
cc     print *,'dqim1(1)=',dqim1
c ... compute the internal forces, fintk, at qk
c       call forces

c     print *,'Check inside itnrml before syseqn'
c       stop
c put qk(i) into big "a(i) as a disp, get out fintk"
        j=0
cc      print *,'neq, neq2, np2 = ',neq,neq2,np2
c       stop
        do 51 j=1,neq
cc          a(j)=qk(j)
                    disp(j)=qk(j)
c c     print *,'a(j) = ',a(j)
c c     print *,'qk(j) = ',qk(j)
 51     continue
c       stop
                    getsolve = 0
        getforce = 1
        getstiff = 0
c     print *,'fintk = ',fintk
c       stop
c     write(6,*)'Check inside itnrml before 4th syseqn'

c                   print *,'b(i) before 4th syseqn=',(b(ic),ic=1,ndof)
c                   print *,'ad before 4th syseqn solve=',(ad(i),i=1,maxdofpc**2)
c                   print *,'an before 4th syseqn solvs=',(an(i),i=1,50)
c                   print *,'disp beforc 4th syseqn in itnrml=',(disp(ic),ic=1,neq)
c                   stop

c                   print *,'Check inside itnml before 4th syseqn'
c                   print *,'!!!!!!!!!!!!!! 4th syseqn inside itnrml !!!!!!!!!!!!!!'
c                   pause
c                   stop
        call syseqn(ne,iplvl,h,ndof,disp,iarc,qk,
     $              ieall,ie,je,iet,jet,jeall,ia,ae,maxdofpe,
     $              neltype,nel,ndofpe,x,y,z,propmat,propsect,lin,
     $              iboundc,ad,an,iperm,iu,ip,iut,dqtot,dqi,dqiin1,
     $              ux,uy,uz,di,it,area,youngm,iarea,
     $           ianal,lumpmass,iskip,itempo2,itnrmlon,exfor,istep,
     $           jallo,
     $           strainel,strainE,strainD,weight,iopt,Anew,Apart,Fin1,
     $           Fin2,VED,VEDold,xlie,SED)

c     $ja,iboundc,ndofped,be,lm,ndofped,ad,ip,tempo1,dm,am,ianal,lumpmass)
c     print *,'getforce = ',getforce
c     write(6,*)'Check inside itnml after 4th syseqn'
c     print *,'Check inside itnml after 4th syseqn'
c                   pause
c                   print *,'ad(_) in itnrml after 4th syseqn=',(ad(ic),ic=1,neq)
c                   pause

c^^^^^^ SETTING ZERO DIAGONAL VALUES TO ONE ^^^^^^^^^^^^^^^^
c       Do 1015 i=neq*2+1,neq*3
c                    Do 1015 i=1,neq
c       If(ad(i) .GT. .000001) GoTo 1015
c       If(ad(i) .GT. 1.e-8) GoTo 1015
c         a(i)=1.0
c                     ad(i)=1.e-8
c                     ad(i)=0.d0
c1015     Continue
c                   print *,'b(i) after new 2nd syseqn=',(b(ic),ic=1,ndof)
c                   print *,'ad after 2nd syseqn solve=',(ad(i),i=1,maxdofpc**2)
c                   print *,'an after 2nd syseqn solvs=',(an(i),i=1,50)
c                   print *,'disp after 4th syseqn in itnrml=',(disp(ic),ic=1,neq)
c                   stop

c                   print *,'qk after 4th syseqn in itnrml=',(qk(ic),ic=1,20)
c                   print *,'dqi after 4th syseqn in itnrml=',(dqi(ic),ic=1,neq)
c                   pause

        do 703 i=1,neq
c           scali(i) = 1.d0/stifi(i,i)
c     c c     print *,'a(neq*2+i) = ',a(neq*2+i)
cc            scali(i) = 1.d0/a(neq*2+i)  !!!!Get diagonal
              scali(i) = 1.d0/ad(i)  !!!!Get diagonal

c                   print *,'scali =',scali(i)
 703        continue
c                   pausc

c                   write(6,*)'scali after 4th syseqn itnrml=',(scali(i),i=1,neq)
c                   print *,'scali after 4th syseqn itnrml=',(scali(i),i=1,neq)
c c     print *,'a=',a
c         stop
cc          call modify
c     print *,'Check after sixth modify'
c c     print *,' "a" after 6th modify = ',a
c     c     pause
          getforce=0 !set back to zero
c c     print *,' "a" after 3rd syseqn to getforce = ',a
c       stop
c** The DO below looks at dqk and assigns 0 to correspond
c** to the boundary conditions which were zeroed out in "a"
c** so if the displacements "qk" stored in the beginning of
c** big "a" equal zero, then "dqk(i)" must be zero.
        Do 49 i=1,neq
c         If(a(i).eq. 0.0)then
cc          If(disp(i).eq. 0.0)then
                    If(iboundc(i) .eq. 1)then
        dqk(i)=0.d0
        endif
c       print *,'dqk(_)=',dqk(i)
 49     continue

cc      print *,'fintk 1a = ',fintk
c       stop

c         If(it .eq. 2)then
cc      print *,'it = ',it
cc      print *,'dqk(i) = ',dqk
c         stop
c         endif
c       print *,'a before resid = ',a
c       stop
c                   print *,'***chk before resid end itnrml***'
c                   pause
c     print *,'exlk bcfore resid=',exlk
c     print *,'exfor before resid=',(exfor(i),i=1,neq)
c     print *,'b before resid=',(b(i),i=1,neq)
c                   pause
c ... compute the residual forces, resk, at qk and exlk
        call resid(dqtot,dqi,dqim1,qk,b,exfor)
c     print *,'resk(_) after 3rd resid in itnrml= ',(resk(ic),ic=1,neq)
```

```
c                       pause
c       print *,'a after 3rd resid in itnrml= ',a
c       stop
c     print *,'exlk after resid= ',exlk

c                       print *,'disp after resid in itnrml=',(disp(ic),ic=1,neq)
c                       print *,'dqtot after 4th sysoqn in itnrml=',(dqtot(ic),ic=1,neq)

c       If(it .eq. 2)then
c     print *,'it = ',it
c                       print *,'disp end itnrml=',(disp(ic),ic=1,ndof)
c                       print *,'b(_) end itnrml=',(b(ic),ic=1,ndof)
c                       print *,'ad end itnrml=',(ad(ic),ic=1,maxdofpe**2)
c                       print *,'an(_) end itnrml=',(an(ic),ic=1,50)
c       stop
c       endif

c                       pause
c       If(it .eq. 2)then
c     print *,'resk(_) = ',(resk(ic),ic=1,neq)
c     print *,'exlk = ',exlk
c                       pause
c       stop
c       endif
c     print *,'stop after resid'
c       stop
c ... test for convergence
cc      print *,'toldis tolfor exlk iconv=',toldis,tolfor,exlk,iconv
c       call test(toldis,tolfor,exlk,iconv)
c                       print *,'chk before test at end itnrml'
c                       print *,'istep=',istep
c                       pause
c                       print *,'Before test 3'

        call test(dqtot,dqim1,qk,exfor,b)
c       call test(disp,dqi,dqim1,qk,exfor)
c     print *, 'iflag after test = ',iflag
c     print *,'iconv after test = ',iconv

c       If(it .eq. 2)then
c     print *,'it = ',it
c                       print *,'iconv when it=2 =',iconv
c                       print *,'iflag when it=2 =',iflag
c       stop
c       endif

c                       pause
c     print *,'stop after test'
c     print *,'a at end of itnrml = ',a

c                       print *,'b(i) end itnrml=',(b(ic),ic=1,ndof)
c                       pause
c                       print *,'ad end itnrml=',(ad(i),i=1,maxdofpe**2)
c                       print *,'an end itnrml=',(an(i),i=1,50)
c                       pause
c                       print *,'disp end itnrml=',(disp(ic),ic=1,neq)
c                       pause
c                       stop
c** Initate a counter for going back to 1002
        iam=iam+1
cc      print *,'iam at end=',iam
c     print *,'exlk at end = ',exlk
c       stop
c       print *,'iconv =',iconv
c       Print *,'it = ',it
c       print *,'itemax =',itemax
c       pause
        go to 1002
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      elseif (iconv .ne. 0 .and. it .gt. itemax) then
c     print *,'it = ',it
c     print *,'Check at iflag = 2'
c       print *,'iconv =',iconv
c       Print *,'it at end of itnrml = ',it
c       print *,'itemax =',itemax
        iflag = 2
      end if
c     print *,'exlk at end of itnrml = ',exlk
c     print *,'Ready to leave itnrml'
c     stop
c
7000  return
      end

C ***********************************************************************
C ***                  SUBROUTINE: STRAIN                             ***
C ***********************************************************************
      subroutine strain(istep,it,dqtot,dqi,dqim1,qk,exfor,ad)
      implicit real*8(a-h,o-z)

      common/RIK/ stifi(60,60),dqk1(5000),
     $     dqk2(5000),fintk(5000),
     $     resk(5000),scali(5000),work1(5000),indx(5000),neq

      common/av/a(1500),jgf,jgsm,np,nbw,np2,nbw2
      common/testing/toldis,tolfor,exlk,iconv

                        dimension dqtot(*),dqi(*),dqim1(*),exfor(*),qk(*),ad(*)

      open(15,file='fca3d2_out.txt',status='unknown')


c**********************************************************************
c** "cris_sdof_in.dat"**************************************************
       appliedforce = exlk * exfor(5)
      if(istep .eq. 1) then
c       write(15,53)
c 53   format(' istep     exlk       appliedforce       a(1)
c     $    a(2)        a(5)       it        ad(3)       ad(4)')

      write(15,53)
 53   format(' istep     exlk       appliedforce       a(1)
     $     a(2)        a(3)        a(4)        a(5)        a(6)
     $     it          ad(4)       ad(5)')

      endif
c       write(15,50) '*',istep,exlk,appliedforce,
c     c     dqtot(1),dqtot(2),dqtot(5),it,
c     c     ad(1),ad(2)
c50    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
c     c         4x,i5,4x,e10.4,3x,e10.4)

      write(15,50) '*',istep,exlk,appliedforce,
     c     dqtot(1),dqtot(2),dqtot(3),dqtot(4),dqtot(5),dqtot(6),it,
     c     ad(4),ad(5)
50    format(a1,2x,i4,5x,e14.6,3x,e10.2,6x,e10.4,6x,e10.4,3X,e10.4,
     c          3x,e10.4,3x, e10.4,3X e10.4,4x,i5,4x,e10.4,3x,e10.4)
      return
      end
```

VITA
Glenn A. Hrinda, P.E.

Department of Civil and Environmental Engineering
Kaufman Hall 135
College of Engineering and Technology
Old Dominion University
Norfolk, VA 23529-0241

Birth Date: March 26, 1961
Birth Place: Honolulu, Hawaii

Education:

Old Dominion University		Norfolk, VA
Doctor of Philosophy in Civil Engineering Special emphasis in Structural Engineering		2009
University of Virginia		Charlottesville, VA
Master of Science in Civil Engineering		1992
Old Dominion University		Norfolk, VA
Bachelor of Science in Civil Engineering		1988

Experience:

NASA Langley Research Center Research Engineer	July 2000 to present
Federal Data Corp. /NASA Structural Engineer	1/1998 - 6/2000
Newport News Shipbuilding Structural Engineer	10/1996 - 12/1997
U.S. Government/Army/Air Force Project Engineer	10/1994 - 10/1996
Sverdrup/Planning Research Corp. Structural Test Engineer	5/1990 - 10/1994
Hampton University Adjunct Professor teaching structural engineering to architect students.	2003-2004